CLASSIFICAÇÃO ASSOCIATIVA SOB DEMANDA

ADRIANO ALONSO VELOSO
Orientador: Wagner Meira Jr.

CLASSIFICAÇÃO ASSOCIATIVA SOB DEMANDA

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Belo Horizonte
Março de 2009

ADRIANO ALONSO VELOSO

Advisor: Wagner Meira Jr.

DEMAND-DRIVEN ASSOCIATIVE CLASSIFICATION

Thesis presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Belo Horizonte

March 2009

Veloso, Adriano Alonso
Demand-Driven Associative Classification / Adriano Alonso Veloso. — Belo Horizonte, 2009
xxxix, 132 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas Gerais
Orientador: Wagner Meira Jr.

1. Data Mining. 2. Machine Learning. 3. Information Retrieval. 4. Digital Libraries.

Abstract

The ultimate goal of machines is to help humans to solve problems. The solutions for such problems are typically programmed by experts, and the machines need only to follow the specified steps to solve the problem. However, the solution of some problems may be too difficult to be explicitly programmed. In such difficult cases, instead of directly programming machines to solve the problem, machines can be programmed to learn the solution. Machine Learning encompasses techniques used to program machines to learn. It is one of the fastest-growing research areas today, mainly motivated by the fact that the advent of improved learning techniques would open up many new uses for machines (i.e., problems for which the solution is hard to program by hand).

A prominent approach to machine learning is to repeatedly demonstrate how the problem is solved, and let the machine learn by example, so that it generalizes some rules about the solution and turn these into a program. This process is known as supervised learning. Specifically, the machine takes matched values of inputs (instantiations of the problem to be solved) and outputs (the solution) and absorb whatever information their relation contains in order to emulate the true mapping of inputs to outputs. When outputs are drawn from a pre-specified and finite set of possibilities, the process is known as classification, which is a major data mining task.

Some classification problems are hard to solve, and motivate this thesis. The key insight that is exploited in this thesis is that a difficult problem can be decomposed into several much simpler sub-problems. This thesis is to show that, instead of directly solving a difficult problem, independently solving its sub-problems by taking into account their particular demands, often leads to improved classification performance. This is shown empirically, by solving real-world problems (for which the solutions are hard to program) using the computationaly efficient algorithms that are presented in this thesis. These problems include categorization of documents and name disambiguation in digital libraries, ranking documents retrieved by search engines, protein functional analysis, revenue optimization, among others. Improvements in classification performance are reported for all these problems (in some cases with gains of more than 100%). Further, theoretical evidence supporting our algorithms is also provided.

Resumo

O objetivo primordial das máquinas é o de ajudar pessoas a resolver problemas. As soluções para tais problemas são geralmente programadas por especialistas, de tal forma que as máquinas precisam apenas seguir os passos que foram especificados no programa. No entanto, as soluções para alguns problemas são muito difíceis de serem programadas explicitamente. Nestes casos, ao invés de programar a máquina para solucionar o problema, a máquina é programada para aprender a solução de tal problema. A Aprendizagem de Máquina compreende o desenvolvimento de técnicas que possam ser usadas para programar máquinas a aprender.

Uma abordagem para a aprendizagem de máquina é demonstrar para a máquina, repetidas vezes, como o problema é solucionado, e simplesmente deixá-la aprender com esses exemplos, de forma que ela possa generalizar regras sobre a solução, e finalmente transformar tais regras em um programa que solucione o problema. Este processo é denominado aprendizagem supervisionada. Neste caso, são fornecidos exemplos de entradas e suas respectivas saídas, de forma que a máquina possa, após absorver o máximo de informação desses exemplos, emular o mapeamento de entradas a saídas. Quando as saídas assumem valores pre-especificados, esse processo é denominado classificação. Classificação é uma das tarefas mais tradicionais em mineração de dados.

Alguns problemas de classificação são extremamente difíceis de solucionar, e motivam esta tese. A intuição explorada nesta tese é que um problema de difícil solução pode ser decomposto em vários sub-problemas mais simples. Esta tese mostra que, solucionar de forma independente sub-problemas mais simples, ao invés de solucionar um problema difícil diretamente, geralmente leva a resultados melhores. Isto é mostrado empiricamente, através da solução de problemas úteis e importantes, usando os algoritmos apresentados nesta tese. Tais problemas incluem categorização de documentos e remoção de ambiguidade em bibliotecas digitais, ordenação de documentos retornados por máquinas de busca, otimização de renda, entre muitos outros. Ganhos em efetividade são reportados em todos estes problemas (em alguns casos com ganhos maiores que 100%). Além disso, apresentamos evidência teórica que suporta nossos algoritmos.

Resumo Estendido

Introdução

A busca por máquinas (ou computadores) capazes de aprender começou por volta de 1950, quando Alan Turing utilizou pela primeira vez o termo "Aprendizagem de Máquina". Turing já imaginava que os computadores poderiam ir além da aritmética. Especificamente, Turing imaginava que os computadores poderiam imitar o processo de aprendizagem dos humanos [Turing, 1951a,b].

Atualmente, o termo "Aprendizagem de Máquina" refere-se à uma das áreas de pesquisa que mais crescem no mundo. Várias classes de problemas referentes à aprendizagem de máquina já foram abordadas. Uma dessas classes engloba os problemas de classificação, nos quais assume-se que o computador terá acesso a exemplos ou demonstrações de como um certo problema é resolvido. Especificamente, esses exemplos são pares de entrada (contendo especificações do problema a ser solucionado) e saída (a solução), e tais entradas e saídas estão relacionadas de alguma maneira desconhecida. Espera-se, caso seja fornecida uma quantidade suficiente de exemplos, que o computador (através da execução de um algoritmo) seja capaz de fornecer uma boa aproximação da solução do problema, ou, mais precisamente, que o computador será capaz de encontrar uma função de mapeamento de entradas para saídas. O grande apelo, nesse caso, é que a solução do problema não precisa ser programada diretamente por um especialista − basta que alguém com um certo entendimento do problema forneça exemplos.

Existem vários algoritmos de classificação. O fator limitante desses algoritmos é o grau de precisão da função de mapeamento que eles fornecem. A dificuldade de um problema de classificação pode fazer com que esses algoritmos obtenham funções pouco precisas, ou que necessitem de um tempo de processamento inaceitável para obter funções que sejam relativamente melhores. Nesta tese exploramos o seguinte conceito: um problema de difícil solução pode ser decomposto em sub-problemas que possuam soluções bem mais simples. Além disso, ao decompor um problema em sub-problemas, demandas específicas desses sub-problemas podem ser levadas em conta

durante o processo de geração das funções de mapeamento.

Propomos vários algoritmos baseados neste conceito intuitivo. Os algoritmos propostos buscam associações entre entradas e saídas que foram fornecidas como exemplo, e as utilizam para reduzir o espaço de busca por funções de mapeamento. Ao invés de encontrar uma única função de mapeamento que fornece uma aproximação da solução do problema, nossos algoritmos decompõem o problema em sub-problemas e produzem várias funções, onde cada função é especificamente produzida levando-se em conta caracteríticas e demandas de cada sub-problema. Este processo é ilustrado pelo exemplo na Figura 1.

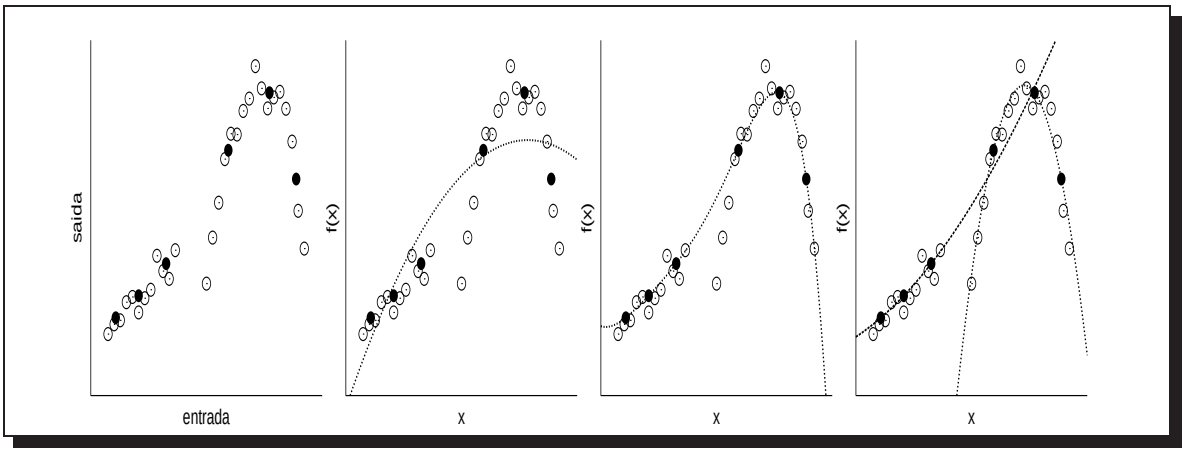

Figura 1. Ilustração do processo de obtenção da função de mapeamento.

O gráfico à esquerda mostra seis pontos negros que foram fornecidos como exemplos. Esses pontos negros compõem o conjunto de treino, que é o conjunto de todos os pontos fornecidos como exemplo. O segundo gráfico mostra uma função de mapeamento que foi construída usando-se todo o conjunto de treino. Os pontos brancos expressam entradas para as quais as respectivas saídas não foram fornecidas ao algoritmo, e que portanto não foram utilizadas durante o processo de construção da função de mapeamento. Tais pontos brancos compõem o conjunto de teste, que é utilizado para avaliar o grau de precisão da função de mapeamento. Sendo assim, a função de mapeamento mostrada na segunda figura não possui um grau de precisão aceitável, uma vez que ela não consegue fornecer saídas corretas para as entradas representadas por pontos brancos. O terceiro figura mostra uma função de mapeamento bem mais complexa, que possui um grau de precisão aceitável. No entanto, o tempo necessário para construir uma função com tal complexidade, pode ser inaceitável. O gráfico à direita mostra duas funções de mapeamento simples, construídas a partir de dois sub-problemas diferentes (i.e., os três primeiros e os três últimos pontos negros). A utilização de múltiplas funções de mapeamento permite que cada função seja construída levando-se em conta

características específicas de cada sub-problema, podendo portanto capturar melhor o relacionamento entre entradas e saídas.

Uma vez que a efetividade dos algoritmos de classificação é medida utilizando-se um conjunto de teste, as funções de mapeamento geradas devem ser especialmente precisas para as entradas no conjunto de teste. Desta forma, os algoritmos propostos nesta tese interpretam cada entrada no conjunto de teste como sendo um sub-problema distinto e produzem uma função de mapeamento para cada uma dessas entradas. Mais especificamente, os exemplos utilizados na construção de uma função para uma determinada entrada no conjunto de teste são apenas aqueles que fornecem alguma informação discriminativa acerca dessa entrada. Considere o exemplo ilustrado na Figura 2. Os três exemplos à esquerda são muito mais informativos para a entrada x_1 do que os três exemplos à direita. Sendo assim, a saída referente à entrada x_1 será dada pela função f_1, que foi construída utilizando-se somente esses três primeiros exemplos. De forma similar, os três exemplos à direita são muito mais informativos para a entrada x_2 do que os três exemplos à esquerda. Portanto, a saída referente à entrada x_2 será dada pela função f_2.

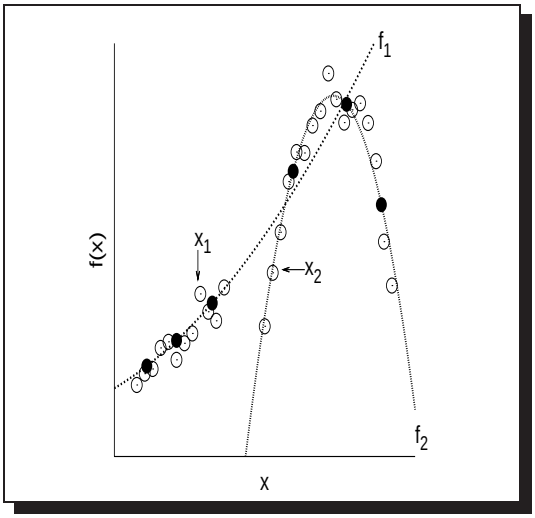

Figura 2. Diferentes funções de mapeamento.

Inicialmente, propomos um algoritmo eficiente do ponto de vista do problema de classificação (i.e., tal algoritmo necessita de poucos exemplos para obter funções de mapeamento precisas). Várias melhorias são discutidas ao longo desta tese e os algoritmos correspondentes são apresentados. Tais algoritmos foram avaliados utilizando-se uma gama de aplicações complexas, tais como categorização e remoção de ambiguidade em bibliotecas digitais, ordenação de documentos retornados por máquinas de busca, otimização de lucro, etc. Ganhos em relação aos melhores algoritmos existentes são reportados para todas essas aplicações.

O Problema de Classificação

A seguir apresentaremos as definições necessárias para que o problema de classificação possa ser formalizado[1].

Conjunto de Treino e Conjunto de Teste

São conjuntos de pares de entrada/saída da forma z=(x_i, y_i). Cada x_i é um registro de tamanho fixo da forma $< a_1, \ldots, a_l >$, onde a_i representa o de valor de um atributo. Cada y_i assume valores provenientes de um conjunto y=$\{c_1, \ldots, c_p\}$ e indica a qual classe o par z pertence. Casos nos quais y_i=? indicam que a classe de z_i é desconhecida. Existe uma distribuição de probabilidade, P$(y|x)$, que governa a relação entre entradas e saídas. Tal distribuição é desconhecida. Pares são divididos em dois conjuntos distintos − o conjunto de treino (denominado $\mathcal{S}$), e o conjunto de teste (denominado $\mathcal{T}$):

$$\begin{aligned} \mathcal{S} &= \{s_1 = (x_1, y_1), \ldots, s_n = (x_n, y_n)\} \\ \mathcal{T} &= \{t_1 = (x_1, ?), \ldots, t_m = (x_m, ?)\} \end{aligned}$$

Algoritmo de Classificação

É um algoritmo que recebe como entrada $\mathcal{S}$ e $\mathcal{T}$, e gera como saída uma função de mapeamento $f_{\mathcal{S}}$, que aproxima, de certa forma, a distribuição P$(y|x)$. Várias funções de mapeamento podem ser produzidas a partir dos exemplos em $\mathcal{S}$. O espaço de hipóteses (denominado $\mathcal{H}$) é o espaço de funções exploradas por um algoritmo de classificação, até que este encontre uma função, denotada como $f_{\mathcal{S}}$, que será usada para mapear entradas à saídas.

Erro Esperado

De acordo com Cucker e Smale [2001] e com Vapnik [1995], o erro esperado de uma função de mapeamento $f_{\mathcal{S}}$ é definido como:

$$\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] = \int_{t=(x,y)} \ell(f_{\mathcal{S}}, t) \mathrm{dP}(y|x)$$

onde $\ell(f_{\mathcal{S}}, t)$ é uma função de perda (i.e., 0-1 loss).

[1] Uma lista com os principais símbolos usados nesta tese pode ser encontrada logo após o índice.

O objetivo principal de um algoritmo de classificação é encontrar uma função $f_{\mathcal{S}}$ para a qual $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$ é garantidamente baixo. No entanto, $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$ não pode ser computado já que $\mathrm{P}(y|x)$ é desconhecida.

Erro Empírico

Embora o erro esperado não possa ser computado, o erro empírico é facilmente calculado usando $\mathcal{S}$:

$$\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\mathcal{S}}, s_i)$$

Generalização

É uma habilidade importante para qualquer algoritmo de classificação. Generalização ocorre quando o erro empírico converge para o erro esperado, com o aumento da quantidade de exemplos fornecidos ao algoritmo, ou seja, $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] \approx \mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$. O erro de generalização (ou risco), denotado como ϵ, é dado por $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] - \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}]$.

Eficiência

Um algoritmo de classificação eficiente é aquele que encontra, em tempo polinomial e com uma quantidade polinomial de exemplos, com probabilidade $(1-\delta)$, uma função $f_{\mathcal{S}} \in \mathcal{H}$, para a qual $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] < \epsilon$, e $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] \approx \mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$.

Técnicas para Aproximação de Função

O problema de classificação é encarado como o problema de se encontrar a função de mapeamento que melhor aproxime $\mathrm{P}(y|x)$. Duas estratégias de aproximação de funções são consideradas nesta tese e são descritas a seguir.

Minimização do Risco Empírico Provavelmente a estratégia de aproximação mais natural é a minimização do risco empírico (ERM − Empirical Risk Minimization): de todas as funções em $\mathcal{H}$, o algoritmo escolhe a função $f_{\mathcal{S}}$ que minimiza $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}]$:

$$\arg\min \left(\frac{1}{n}\sum_{i=1}^{n} \ell(f_{\mathcal{S}}, s_i) \right), \forall f_{\mathcal{S}} \in \mathcal{H} \tag{1}$$

No entanto, a minimização do risco empírico não garante generalização. Mais especificamente, a minimização do erro empírico não implica necessariamente na mini-

mização do erro esperado. Uma condição suficiente para generalização de algoritmos baseados em ERM á a estabilidade de $f_{\mathcal{S}}$ [Mukherjee et al., 2006; Poggio et al., 2004].

A estabilidade mede a diferença, β_{s_i}, dos erros empíricos no par $s_i \in \mathcal{S}$ quando consideramos a função $f_{\mathcal{S}}$ obtida utilizando-se todo o conjunto de treino $\mathcal{S}$ e a função $f_{\mathcal{S}-s_i}$ obtida quando não levamos o par s_i em consideração. Em outras palavras, se o conjunto de treino $\mathcal{S}$ é perturbado pela remoção do par s_i, e se a função $f_{\mathcal{S}}$ não diverge muito da função $f_{\mathcal{S}-s_i}$, então $f_{\mathcal{S}}$ é estável. A função $f_{\mathcal{S}}$ é β-estável se:

$$\forall s_i \in \mathcal{S}, |f_{\mathcal{S}}(s_i) - f_{\mathcal{S}-s_i}(s_i)| \leq \beta \tag{2}$$

O menor valor de β em 2 indica a estabilidade de $f_{\mathcal{S}}$. O menor valor de β é a maior variação no par s_i. Dessa forma, a função $f_{\mathcal{S}}$ mostrada na Figura 3, é obtida através da minimização do risco empírico usando-se $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5\}$. De forma similar, a função $f_{\mathcal{S}-s_2}$ é obtida através da minimização do risco empírico usando-se $\{\mathcal{S} - s_2\}$ e a função $f_{\mathcal{S}-s_5}$ ŕ obtida através da minimização do risco empírico usando-se $\{\mathcal{S} - s_5\}$. A diferença no par s_2, β_{s_2}, é baixa. A diferença no par s_5, β_{s_5}, é alta. Consequentemente, $f_{\mathcal{S}}$ é β_{s_5}-estável. A função $f_{\mathcal{S}}$ é estável se $\beta = O(\frac{1}{n})$.

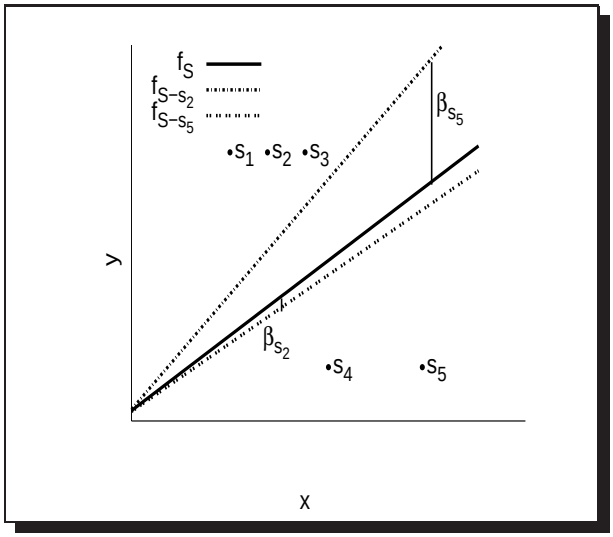

Figura 3. Minimização do Risco Empírico

Bousquet e Eliseef [2002] mostraram que o erro esperado pode ser estimado pelo erro empírico e a estabilidade da função $f_{\mathcal{S}}$, da seguinte forma:

$$\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] \leq \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] + \left(\beta + (4n\beta + 1) \times \sqrt{\frac{\ln \frac{1}{\delta}}{2n}}\right) \tag{3}$$

Sendo assim, a função $f_{\mathcal{S}}$ que minimiza $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$ pode ser encontrada aplicando a Inequação 3 para cada função candidata em $\mathcal{H}$.

Minimização do Risco Estrutural A minimização do risco estrutural (SRM − Empirical Risk Minimization) representa uma escolha entre a complexidade da função e o seu respectivo erro empírico. Funções simples podem fornecer erros empíricos altos, enquanto funções complexas podem fornecer erros empíricos baixos. Sendo assim, de todas as possíveis funções em $\mathcal{H}$, algoritmos baseados em SRM selecionam a função $f_{\mathcal{S}}$ que oferece o melhor balanço entre complexidade e erro empírico.

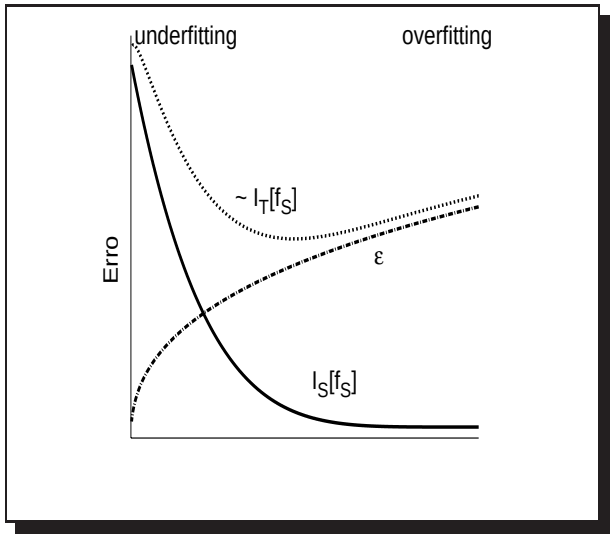

Figura 4. Minimização do Risco Estrutural

Uma estrutura é um conjunto de classes de funções $\mathcal{F}_i$, tal que $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots$, onde funções em $\mathcal{F}_1$ são mais simples (i.e., têm menor complexidade) que funções em $\mathcal{F}_2 - \mathcal{F}_1$, e assim sucessivamente. Já que tais classes de funções são aninhadas, o erro empírico tende a diminuir com o aumento da complexidade.

Uma medida de complexidade amplamente usanda é a chamada dimensão VC [Vapnik and Chervonenkis, 1971; Blumer et al., 1989] de uma função $f_{\mathcal{S}}$, que aqui é denotada como $d_{f_{\mathcal{S}}}$. A dimensão VC mede o poder de expressão de uma função verificando o quão complicada essa função pode ser. Foi mostrado em [Guyon et al., 1992] que o erro esperado pode ser estimado pelo erro empírico e pela complexidade de $f_{\mathcal{S}}$, da seguinte forma:

$$\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] \leq \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] + \sqrt{\frac{d_{f_{\mathcal{S}}}\left(\ln \frac{2n}{d_{f_{\mathcal{S}}}} + 1\right) - \ln \frac{\delta}{4}}{n}} \tag{4}$$

O formato induzido por esta inequação é mostrado na Figura 4. A minimização do risco estrutural busca encontrar a função $f_{\mathcal{S}}$ que seja simples e que forneça o menor erro empírico.

Classificação Associativa

O espaço de hipóteses, $\mathcal{H}$, pode conter uma quantidade infinita de funções de mapeamento. Produzir funções aleatoriamente, na esperança de encontrar uma função que aproxime bem $\mathrm{P}(y|x)$, não é de forma alguma uma estratégia eficiente. Felizmente, existem várias estratégias mais eficientes. Uma dessas estratégias é explorar associações entre entradas e saídas (que nesse caso são denominadas classes). Tais associações são usadas para produzir funções de mapeamento precisas. Esta estratégia é geralmente denominada classificação associativa. A função de mapeamento é composta por regras $\mathcal{X} \rightarrow c_j$, que indicam uma associação entre $\mathcal{X}$ (que é um conjunto de valores de atributos, também chamados de características) e uma classe $c_j \in y$.

Regras de Decisão

São implicações da forma $\mathcal{X} \rightarrow c_j$, onde $\mathcal{X}$ é um conjunto de características e $c_j \in y$ é uma classe. Tais implicações são mapeamentos locais de entrada para saídas, que são extraídos de $\mathcal{S}$. Sendo assim, uma regra $\mathcal{X} \rightarrow c_j$ só existe se as caracteríticas em $\mathcal{X}$ estiverem presentes em $\mathcal{S}$. Alguns conceitos importantes acerca das regras de decisão são apresentados a seguir.

Uma regra de decisão só é interessante caso a informação fornecida por ela seja confiável. O suporte de uma regra $\mathcal{X} \rightarrow c_j$, que é denotado por $\sigma(\mathcal{X} \rightarrow c_j)$, é uma indicação importante do quão confiável é a informação fornecida pela regra. Formalmente, o suporte é definido como:

$$\sigma(\mathcal{X} \rightarrow c_j) = \frac{|(x_i, y_i)| \in \mathcal{S} \text{ tal que } \mathcal{X} \subseteq x_i \text{ e } c_j = y_i}{n} \tag{5}$$

Uma regra $\mathcal{X} \rightarrow c_j$ só é interessante caso $\mathcal{X}$ e c_j sejam associados de alguma forma. A confiança da regra $\mathcal{X} \rightarrow c_j$, que é denotada por $\theta(\mathcal{X} \rightarrow c_j)$, é uma indicação de quão forte é a associação entre $\mathcal{X}$ e c_j. Formalmente, a confiança é definida como:

$$\theta(\mathcal{X} \rightarrow c_j) = \frac{|(x_i, y_i)| \in \mathcal{S} \text{ tal que } \mathcal{X} \subseteq x_i \text{ e } c_j = y_i}{|(x_i, y_i)| \in \mathcal{S} \text{ tal que } \mathcal{X} \subseteq x_i} \tag{6}$$

Geralmente, a complexidade de uma regra de decisão $\mathcal{X} \rightarrow c_j$ é dada pelo seu tamanho, ou seja, pelo número de características incluídas na regra (i.e., $|\mathcal{X}|$).

Finalmente, uma regra $\mathcal{X} \rightarrow c_j$ só é aplicável a uma entrada $x_i \in \mathcal{T}$, caso $\mathcal{X} \subseteq x_i$. Caso contrário, a regra é considerada inútil para fins de prever a classe de x_i.

A seguir novos algoritmos baseados em classificação associativa serão apresentados.

EAC-SR (acrônimo derivado de "*eager associative classification using a single rule*") – É o algoritmo mais simples a ser apresentado nesta tese. Dada uma entrada x_i, esse algoritmo retorna a classe prevista pela regra $\mathcal{X} \rightarrow c_j$ (com $\mathcal{X} \subseteq x_i$) que possua o maior valor de confiança. Embora seja um algoritmo muito simples, pode-se demonstrar que EAC-SR é um algoritmo eficiente do ponto de vista do problema de classificação. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 1 (página 33 da versão completa da tese).

Várias melhorias são propostas a partir do algoritmo EAC-SR. Tais melhorias levam à elaboração dos outros algoritmos descritos nesta tese.

EAC-MR (acrônimo derivado de "*eager associative classification using multiple rules*") – Este algoritmo utiliza múltiplas regras para prever a classe de uma entrada. Cada regra $\mathcal{X} \rightarrow c_j$ é interpretada como um voto dado por $\mathcal{X}$ à classe c_j. O peso do voto é dado por $\theta(\mathcal{X} \rightarrow c_j)$. A pontuação de uma classe c_j, referente à entrada x_i, é definida como:

$$s(x_i, c_j) = \frac{\sum_{r \in \mathcal{R}_{c_j}^{x_i}} \theta(r)}{\mid \mathcal{R}_{c_j}^{x_i} \mid} \tag{7}$$

onde $\mathcal{R}_{c_j}^{x_i}$ é o conjunto de regras que são aplicáveis para a entrada x_i, e que prevêem a classe c_j. A probabilidade da classe c_j ser a saída correta da entrada x_i, denotada por $\hat{p}(c_j|x_i)$, é dada por:

$$\hat{p}(c_j|x_i) = \frac{s(x_i, c_j)}{\sum_{k=1}^{p} s(x_i, c_k)} \tag{8}$$

onde p é o número de possíveis classes em $\mathcal{S}$. Finalmente, o algoritmo EAC-MR retorna a classe com maior probabilidade de ser a saída para x_i. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 2 (página 35 da versão completa da tese).

EAC-MR-ERM (acrônimo derivado de "*empirical risk minimization*") – Funções de mapeamento construídas a partir de regras complexas (i.e., regras que contêm muitas características), fornecem baixo erro empírico. No entanto, como discutido anteriormente, tais funções só serão efetivas caso sejam estáveis. O algoritmo EAC-MR-ERM utiliza a Inequação 3 para encontrar uma função de mapeamento estável, e que ao mesmo tempo forneça um erro empírico baixo. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 3 (página 37 da versão completa da tese).

EAC-MR-SRM (acrônimo derivado de "*structural risk minimization*") – O algoritmo EAC-MR-SRM utiliza a Inequação 4 de forma a escolher funções de mapeamento que sejam simples e que também sejam capazes de fornecer baixo erro empíirico. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 4 (página 39 da versão completa da tese).

Principais Resultados

Avaliamos a efetividade dos algoritmos propostos em um importante problema de classificação denominado categorização de documentos. Para tanto utilizamos uma coleção de documentos extraídos da biblioteca digital da ACM (Association for Computing Machinery). São quase 7.000 documentos, onde cada documento pode ser enquadrado em uma de 8 categorias. Partes dos resultados a serem apresentados podem ser encontradas em [Veloso et al., 2006a].

Vários algoritmos diferentes são empregados para efeitos de comparação. A Tabela 1 mostra os resultados obtidos por cada algoritmo avaliado. Todos os resultados são estatisticamente significativos de acordo com o teste-T com 95% de confiança. Os melhores resultados, incluindo empates estatísticos, são mostrados em negrito. O algoritmo Multi-Kernel [Joachims et al., 2001] é o que oferece os melhores resultados. Em contrapartida, ele necessita de um tempo de processamento muito alto. O algoritmo SVM parece ser o aquele que oferece o melhor custo-benefício entre effetividade e rapidez. Os algoritmos EAC-SR, EAC-MR, EAC-MR-ERM, e EAC-MR-SRM não foram efetivos para esta coleção. A principal causa é a dificuldade de extrair regras com baixo suporte. Tais regras são importantes para aumentar a precisão da função de mapeamento. A seguir, vamos apresentar algoritmos que extraem regras sob demanda, e que, portanto, conseguem ser mais efetivos.

Algoritmos	$MicF_1$	$MacF_1$	Ganhos (%) relativos ao baseline		Tempo de Execução
			$MicF_1$	$MacF_1$	
Amsler (baseline)	0,832	0,783	–	–	1.251 segundos
EAC-MR	0,766	0,692	-0,079	-0,115	2.350 segundos
EAC-MR-ERM	0,789	0,736	-0,051	-0,060	2.921 segundos
EAC-MR-SRM	0,812	0,767	-0,024	-0,020	2.419 segundos
kNN	0,833	0,774	0,001	-0,011	83 segundos
SVM	0,845	0,810	0,016	0,035	1.932 segundos
Bayesian	**0,847**	0,796	0,019	0,016	8.281 segundos
Multi-Kernel	**0,859**	**0,812**	0,032	0,037	14.894 segundos

Tabela 1. Efetividade de diferentes algoritmos.

Classificação Associativa Sob Demanda

A classificação associativa sob demanda baseia-se na intuição de que um problema pode ser decomposto em sub-problemas mais simples, os quais, por sua vez, podem ser resolvidos independentemente. A seguir, tornaremos tal intuição mais precisa.

Projeção

As projeções formam o conceito chave por trás da decomposição de um problema em sub-problemas. Especificamente, dada uma entrada $x_i \in \mathcal{T}$, o conjunto de treinamento, $\mathcal{S}$, é projetado de forma que seja possível extrair apenas regras $\mathcal{X} \rightarrow c_j$ para as quais $\mathcal{X} \subseteq x_i$. Tal procedimento geralmente reduz significativamente a quantidade de regras geradas. A projeção, que é denotada por $\mathcal{S}^{x_i}$, é obtida através da filtragem de características que não carregam informação discriminatória acerca de x_i. Dessa forma, cada projeção $\mathcal{S}^{x_i}$ é um sub-problema de $\mathcal{S}$ (i.e., $\mathcal{S}^{x_i} \subseteq \mathcal{S}$). A seguir apresentaremos algoritmos que produzem múltiplas funções de mapeamento − mais precisamente, uma função de mapeamento, $f_{\mathcal{S}}^{x_i}$, é produzida a partir de cada sub-problema $\mathcal{S}^{x_i}$. A função $f_{\mathcal{S}}^{x_i}$ é produzida de forma a fornecer uma aproximação especialmente precisa para a entrada x_i.

LAC-SR (acrônimo derivado de "*lazy associative classification using a single rule*") − Dada uma entrada $x_i \in \mathcal{T}$, esse algoritmo extrai regras de decisão a partir de cada projeção $\mathcal{S}^{x_i}$. Em seguida, ele retorna a classe prevista pela regra que possua o maior valor de confiança.

LAC-MR (acrônimo derivado de "*lazy associative classification using multiple rules*") − Dada uma entrada $x_i \in \mathcal{T}$, esse algoritmo extrai regras de decisão a partir de cada projeção $\mathcal{S}^{x_i}$. Em seguida, ele utiliza múltiplas regras para prever a classe de x_i. Cada regra $\mathcal{X} \rightarrow c_j$ é interpretada como um voto dado por $\mathcal{X}$ à classe c_j. O peso do voto é dado por $\theta(\mathcal{X} \rightarrow c_j)$. A pontuação de uma classe c_j, referente à entrada x_i, é definido pela Equação 7. A probabilidade da classe c_j ser a classe correta da entrada x_i é definida pela Equação 8. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 5 (página 51 da versão completa da tese).

LAC-MR-ERM (acrônimo derivado *empirical risk minimization*) − Cada sub-problema $\mathcal{S}^{x_i}$ pode demandar funções com diferentes níveis de complexidade. O algoritmo LAC-MR-ERM utiliza a Inequação 3 para encontrar uma função de mapeamento estável para o sub-problema $\mathcal{S}^{x_i}$ e que ao mesmo tempo forneça um erro empírico baixo

em $\mathcal{S}^{x_i}$. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 6 (página 53 da versão completa da tese).

LAC-MR-SRM (acrônimo derivado de "*structural risk minimization*") – Cada sub-problema $\mathcal{S}^{x_i}$ pode demandar funções com diferentes níveis de complexidades. O algoritmo LAC-MR-SRM utiliza a Inequação 4 para encontrar uma função de mapeamento simples para o sub-problema $\mathcal{S}^{x_i}$, e que ao mesmo tempo forneça um erro empírico baixo em $\mathcal{S}^{x_i}$. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 7 (página 54 da versão completa da tese).

Principais Resultados

Avaliamos a efetividade dos algoritmos propostos em um importante problema de classificação denominado categorização de documentos. Novamente, utilizamos uma coleção de documentos extraídos da biblioteca digital da ACM (Association for Computing Machinery). Tal coleção já foi descrita anteriormente. Partes dos resultados a serem apresentados podem ser encontradas em [Veloso et al., 2006a].

Vários algoritmos diferentes são empregados para efeitos de comparação. A Tabela 2 mostra os resultados obtidos por cada algoritmo avaliado. Todos os resultados são estatisticamente significativos de acordo com o test-T com 95% de confiança. Os melhores resultados, incluindo empates estatísticos, são mostrados em negrito. Os algoritmos baseados na classificação associativa sob demanda oferecem os melhores resultados. Além disso, eles também estão entre os mais rápidos. Estes resultados demonstram as vantagens da classificação associativa sob demanda.

Algoritmos	$MicF_1$	$MacF_1$	Ganhos (%) relativos ao baseline		Tempo de Execução
			$MicF_1$	$MacF_1$	
Amsler (baseline)	0,832	0,783	–	–	1.251 segundos
kNN	0,833	0,774	0,001	-0,011	83 segundos
SVM	0,845	0,810	0,016	0,035	1.932 segundos
Bayesian	0,847	0,796	0,019	0,016	8.281 segundos
Multi-Kernel	0,859	0,812	0,032	0,037	14.894 segundos
LAC-MR	**0,862**	0,814	0,036	0,040	257 segundos
LAC-MR-ERM	**0,868**	**0,833**	0,043	0,064	504 segundos
LAC-MR-SRM	**0,873**	**0,839**	0,049	0,071	342 segundos

Tabela 2. Efetividade de diferentes algoritmos.

Extensões à Classificação Associativa Sob Demanda

A seguir aprentaremos extensões aos algoritmos baseados na classificação associativa sob demanda. Tais extensões visam aumentar a gama de aplicações beneficiadas pela técnica. Os algoritmos a serem apresentados, e os resultados obtidos por esses algoritmos, são discutidos em um nível de detalhamento maior na versão completa da tese.

Classificação Multi-Rotulada

Frequentemente, várias saídas (i.e., rótulos) estão relacionadas a uma mesma entrada. Dois algoritmos para classificação multi-rotulada sob demanda são apresentados a seguir.

LAC-MR-IO (acrônimo derivado de "*independent outputs*") − Este algoritmo é similar ao algoritmo LAC-MR. A diferença é a utilização de um novo parâmetro, Δ_{min} ($0 \leq \Delta_{min} \leq 0.5$). Nesse caso, para uma entrada $x_i \in \mathcal{T}$, se $\hat{p}(c_j|x_i) \geq \Delta_{min}$, então a saída c_j é reconhecida como sendo uma das saídas associadas à entrada x_i. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 8 (página 63 da versão completa da tese).

LAC-MR-CO (acrônimo derivado de "*correlated outputs*") − Este algoritmo explora possíveis correlações entre diferentes saídas, de forma a produzir funções de mapeamento ainda melhores. Nesse caso, ao se identificar uma saída para uma entrada $x_i \in \mathcal{T}$, tal saída é posteriormente tratada como uma característica, e portanto pode ser inserida no antecedente das regras. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 9 (página 64 da versão completa da tese).

Resumo dos Resultados Os algoritmos propostos foram avaliados em aplicações reais. Especificamente, o algoritmo LAC-MR-CO oferece os melhores resultados. Ganhos de até 24% são obtidos quando LAC-MR-CO é comparado aos algoritmos propostos em [Elisseeff and Weston, 2001; Schapire and Singer, 2000; Comité et al., 2003]. Resultados mais detalhados podem ser encontrados em Veloso et al. [2007a].

Classificação Multi-Métrica

Existem várias métricas que podem ser usadas para quantificar a associação entre $\mathcal{X}$ e c_j (i.e., confiança, correlação etc.). Algoritmos que utilizam métricas diferentes frequentemente geram resultados diferentes. Classificação multi-métrica envolve a

combinação dos resultados retornados por algoritmos que utilizam métricas diferentes. Estes algoritmos são chamados de algoritmos-base. Três algoritmos para classificação multi-métrica sob demanda são apresentados a seguir.

LAC-MR-SD (acrônimo derivado de "*self-delegation*")– Este algoritmo escolhe, por conta própria, qual algoritmo-base será utilizado para aproximar a saída de uma entrada $x_i \in \mathcal{T}$. A escolha é baseada nos valores de $\hat{p}(c_j|x_i)$ gerados por cada algoritmo-base. Especificamente, o algoritmo-base que produz a função que retorna o maior valor de $\hat{p}(c_j|x_i)$ é o escolhido, e c_j é a saída retornada. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 10 (página 73 da versão completa da tese).

LAC-MR-OC (acrônimo derivado de "*output-centric*") – Este algoritmo utiliza um meta-classificador, que escolhe qual algoritmo-base que será utilizado para aproximar a saída de uma entrada $x_i \in \mathcal{X}$. A escolha é baseada na competência de cada algoritmo-base com relação à saída que será retornada. A intuição é que alguns algoritmos-base fornecem bons resultados quando prevêem certas saídas, mas não fornecem bons resultados quando prevêem outras saídas. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 12 (página 75 da versão completa da tese).

LAC-MR-IC (acrônimo derivado de "*input-centric*") – Este algoritmo utiliza um meta-classificador, que escolhe qual algoritmo-base que será utilizado para aproximar a saída de uma entrada $x_i \in \mathcal{X}$. A escolha é baseada na competência de cada algoritmo-base com relação às características de cada entrada. A intuição é que alguns algoritmos-base fornecem bons resultados apenas para certas entradas. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 13 (página 76 da versão completa da tese).

Resumo dos Resultados Os algoritmos propostos foram avaliados em aplicações reais. Especificamente, o algoritmo LAC-MR-IC oferece os melhores resultados. Ganhos de mais de 8,5% são obtidos quando LAC-MR-IC é comparado aos algoritmos propostos em [Ortega et al., 2001]. Resultados mais detalhados podem ser encontrados em Veloso et al. [2009c,d].

Classificação Calibrada

Algumas aplicações necessitam que as probabilidades $\hat{p}(c_j|x_i)$ sejam extremamente precisas, ou seja, o valor da aproximação $\hat{p}(c_j|x_i)$ deve ser o mais próximo possível do valor

real. Quando isso acontece, diz-se que o algoritmo está calibrado. Dois algoritmos para classificação calibrada sob demanda são apresentados a seguir.

LAC-MR-NC (acrônimo derivado de "*naive calibration*") – Este algoritmo utiliza uma suavização baseada em histogramas. Especificamente, a acurácia das previsões em cada histograma é usada para calibrar as probabilidades $\hat{p}(c_j|x_i)$. A quantidade e os limites dos histogramas são fornecidos pelo usuário.

LAC-MR-EM (acrônimo derivado de "*entropy minimization*") – Este algoritmo utiliza uma suavização baseada em histogramas. Especificamente, a acurácia das previsões em cada histograma é usada para calibrar as probabilidades $\hat{p}(c_j|x_i)$. A quantidade e os limites dos histogramas são obtidos automaticamente, através de um processo de minimização da entropia em cada histograma. Tal processo cria histogramas novos até que o ganho de informação obtido com a criação de um histograma seja menor que o tamanho da descrição mínima (MDL) desse histograma [Rissanen, 1978].

Resumo dos Resultados Os algoritmos propostos foram avaliados em aplicações reais. Especificamente, o algoritmo LAC-MR-EM oferece os melhores resultados. Ganhos superiores a 17,5% são obtidos quando LAC-MR-EM é comparado com outros algoritmos propostos em [Platt, 1999; Cestnik, 1990; Zadrozny and Elkan, 2001]. Resultados mais detalhados podem ser encontrados em Veloso et al. [2008b, 2009b].

Auto-Treinamento

Algumas aplicações possuem características que dificultam a produção de exemplos de treino. Entre tais dificuldades, destacamos a ambiguidade entre as saídas, que pode trazer confusão ao especialista. Um algoritmo capaz de realizar auto-treinamento é apresentado a seguir.

LAC-MR-ST (acrônimo derivado de "*self-training*")– Este algoritmo utiliza uma nova técnica de auto-treinamento, onde (1) a falta de informação, e (2) a certeza nas previsões, são usadas para a produção automática de novos exemplos de treino.

Resumo dos Resultados Os algoritmos propostos foram utilizados na remoção de ambiguidade de nomes em bibliotecas digitais. Especificamente, o algoritmo LAC-MR-ST oferece resultados similares aos resultados obtidos por algoritmos que têm acesso a informação privilegiada [Han et al., 2005]. Resultados mais detalhados podem ser encontrados em Veloso et al. [2009a].

Regressão Ordinal

Algumas aplicações precisam ordenar as entradas de acordo com algum critério pré-estabelecido. Este é o caso, por exemplo, de muitas das aplicações de Recuperação de Informação, onde documentos devem ser ordenados de acordo com a respectiva relevância para a consulta. Um algoritmo capaz de ordenar as entradas $x_i \in \mathcal{T}$, de acordo com suas relevâncias, é apresentado a seguir.

LAC-MR-OR (acrônimo derivado de "*ordinal regression*") – Este algoritmo produz probabilidades $\hat{p}(c_j|x_i)$, onde $x_i \in \mathcal{T}$ e c_j é uma possível classe de relevância. Essas probabilidades são combinadas linearmente, de acordo com a Equação 9, de forma que o valor $rank(x_i)$ obtido através dessa operação possa ser usado para fornecer a posição da entrada x_i. Os passos principais seguidos por este algoritmo estão descritos no Algoritmo 17 (página 108 da versão completa da tese).

$$rank(x_i) = \sum_{j=0}^{p} \left(c_j \times \hat{p}(c_j|x_i) \right) \tag{9}$$

Resumo dos Resultados Os algoritmos propostos foram avaliados em aplicações reais. Especificamente, o algoritmo LAC-MR-OR oferece os melhores resultados. Os ganhos fornecidos pelo algoritmo LAC-MR-OR variam de 6,6% a 42%, quando ele é comparado com os algoritmos propostos em [Yue et al., 2007; Tsai et al., 2007; Freund et al., 2003; Joachims, 2002; Xu and Li, 2007; Cao et al., 2007]. Resultados mais detalhados podem ser encontrados em Veloso et al. [2008a].

Conclusões

Nesta tese tratamos uma classe de problemas que são amplamente conhecidos como problemas de classificação. Dado um conjunto de entradas e suas respectivas saídas, que são de alguma forma relacionadas entre si, o objetivo é produzir uma função de mapeamento capaz de aproximar a relação entre entradas e saídas, de forma que essa função seja utilizada para prever saídas para entradas arbitrárias. Propusemos vários algoritmos de classificação. Desses algoritmos, mostramos que o mais simples deles é eficiente do ponto de vista do problema de classificação. Aplicamos melhorias diversas a esse algoritmo, que resultaram na criação de vários outros algoritmos, bem mais sofisticados. Esse processo de melhoria contínua culminou na criação de algoritmos baseados na classificação associativa sob demanda. Utilizamos problemas reais para mostrar que esses algoritmos produzem funções de mapeamento com alto grau de precisão.

A intuição chave por trás desses algoritmos é a de que um problema complexo pode ser decomposto em vários sub-problemas bem mais simples e que tais sub-problemas podem ser resolvidos independentemente. Finalmente, propusemos extensões a esses algoritmos, de forma que eles possam solucionar problemas relacionados ao problema de classificação original.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

x: the space of inputs.

y: the discrete space of outputs.

a: the space of attributes.

$\mathcal{H}$: the hypothesis space.

x_i: an arbitrary input.

c_i: an arbitrary output (or label).

γ_i: an arbitrary metric.

a_i: an arbitrary attribute-value (or feature).

s_i: an arbitrary pair in $\mathcal{S}$.

z_i: an arbitrary pair in $\mathcal{T}$.

$\mathcal{S}$: an arbitrary training data.

$\mathcal{T}$: an arbitrary test set.

$\mathcal{X}$: an arbitrary set of features.

$\mathcal{X} \rightarrow c_i$ or r refer to an arbitrary decision rule.

$\mathcal{R}$: an arbitrary set of rules.

$\mathcal{R}^{x_i}$: an arbitrary set of rules matching x_i.

$\mathcal{R}^{c_i}$: an arbitrary set of rules predicting c_i.

m: the number of pairs in the test set.

n: the number of pairs in the training data.

p: the number of outputs.

l may refer to the number of attributes of an input, or to the left boundary of a bin.

q: the number of features in $\mathcal{S}$.

$f_{\mathcal{S}}$: a discrete approximation of $\mathrm{P}(y|x)$.

$f_{\mathcal{S}}^{x_i}$: a discrete approximation of $\mathrm{P}(y|x)$, which is specially accurate at input x_i.

$\sigma(\mathcal{X} \rightarrow c_i)$: the support of a rule.

$\theta(\mathcal{X} \rightarrow c_i)$: the confidence of a rule.

β: the stability of a function.

σ_{min}: the minimum support threshold.

Δ_{min}: a threshold indicating the predicted outputs in multi-label classification.

ϕ_{min}: a threshold indicating that the prediction is reliable.

τ: the degree of calibration of a classification algorithm.

Chapter 1

Introduction

Learning is a fundamental ability of many living organisms. It leads to the development of new skills, values, understanding, and preferences. Improved learning capabilities catalyze the evolution and may distinguish entire species with respect to the activities they are able to perform. The importance of learning is, thus, beyond question. Learning covers a broad range of tasks. Some tasks are particularly interesting because they can be mathematically modeled. This makes natural to wonder whether machines might be made, or programmed, to learn.

A deep understanding of how to program machines to learn is still distant, but it would be of great impact because it would increase the spectrum of problems that machines can solve. Candidate problems range between two extremes: structured problems for which the solution is totally defined (and thus are easily programmed by humans [Hutter, 2002]), and random problems for which the solution is completely undefined (and thus cannot be programmed). Problems in the vast middle ground have solutions that cannot be well defined and are, thus, inherently hard to program. Machine Learning is the way to handle this vast middle ground and many tedious and difficult hand-coding tasks would be replaced by automatic learning methods.

A prominent approach to machine learning is to provide to the machine examples demonstrating how the problem is solved. These examples are paired values of inputs (instantiations of the problem to be solved) and outputs (the corresponding solution). Inputs and outputs are related somehow, but this relationship is unknown. The machine must generalize rules about this relationship and turn these rules into a program. This program will predict the outputs associated with inputs for which the solution is unknown. When the solution assumes pre-defined and finite values (which are called classes), this process is known as classification. Classification is a major task in predictive data mining [Witten and Frank, 2005]. According to Wu et al. [2008], six out of the ten most influential data mining algorithms are classification algorithms.

The relationship between inputs and outputs may be expressed as a mapping function, which takes an input and provides the corresponding output. Since this function is unknown, the classification problem can be essentially stated as a function approximation problem: given as examples some inputs for which the outputs (i.e., the classes) are known, the goal is to extrapolate the (unknown) outputs associated with yet unseen inputs as accurately as possible. Several classification algorithms follow this function approximation paradigm [Evgeniou et al., 2000; Poggio and Girosi, 1998; Rahimi and Recht, 2008]. These algorithms usually rely on a single mapping function to approximate the target function (i.e., the relationship between inputs and outputs). This single function is selected from a set of candidate functions and is the one which is most likely to provide the best available approximation to the target function. This implies that such single function will be used to approximate the target function over the full space of inputs. This is not necessarily a good strategy, because:

- the set of possible functions might not contain a good approximation of the target function for the full input space;
- the use of a single function to approximate the target function over the full space of inputs tends to be good on average, but it may fail on some particular regions (or ranges) of the input space.

Figure 1.1 illustrates the function approximation process for a classification problem. The left-most graph on the top (middle and right) shows the target function, where each point represents an input-output pair. The black points are given as examples to the classification algorithm, which uses them to build the mapping function. The white points are used to assess the accuracy of this function. Different mapping functions are shown. Graphs on the top show mapping functions that do not provide a good approximation for the target function. Graphs on the bottom show mapping functions that fit well the target function, although they still fail on some particular regions of the input space.

The limiting factor of classification algorithms is the accuracy of the mapping functions they can provide in a reasonable time. Dramatic gains cannot be achieved through minor algorithmic modifications, but require the introduction of new strategies and approaches. The key approach we exploited in this thesis, in order to enhance the accuracy of classification algorithms, is to decompose a hard classification problem into much easier sub-problems, where each sub-problem is defined by inputs that are similar somehow (i.e., a range of the input space). Then, a specific mapping function for each sub-problem is built independently from each other, on a demand-driven basis, according to particularities of each sub-problem. This strategy leads to a finer-grained

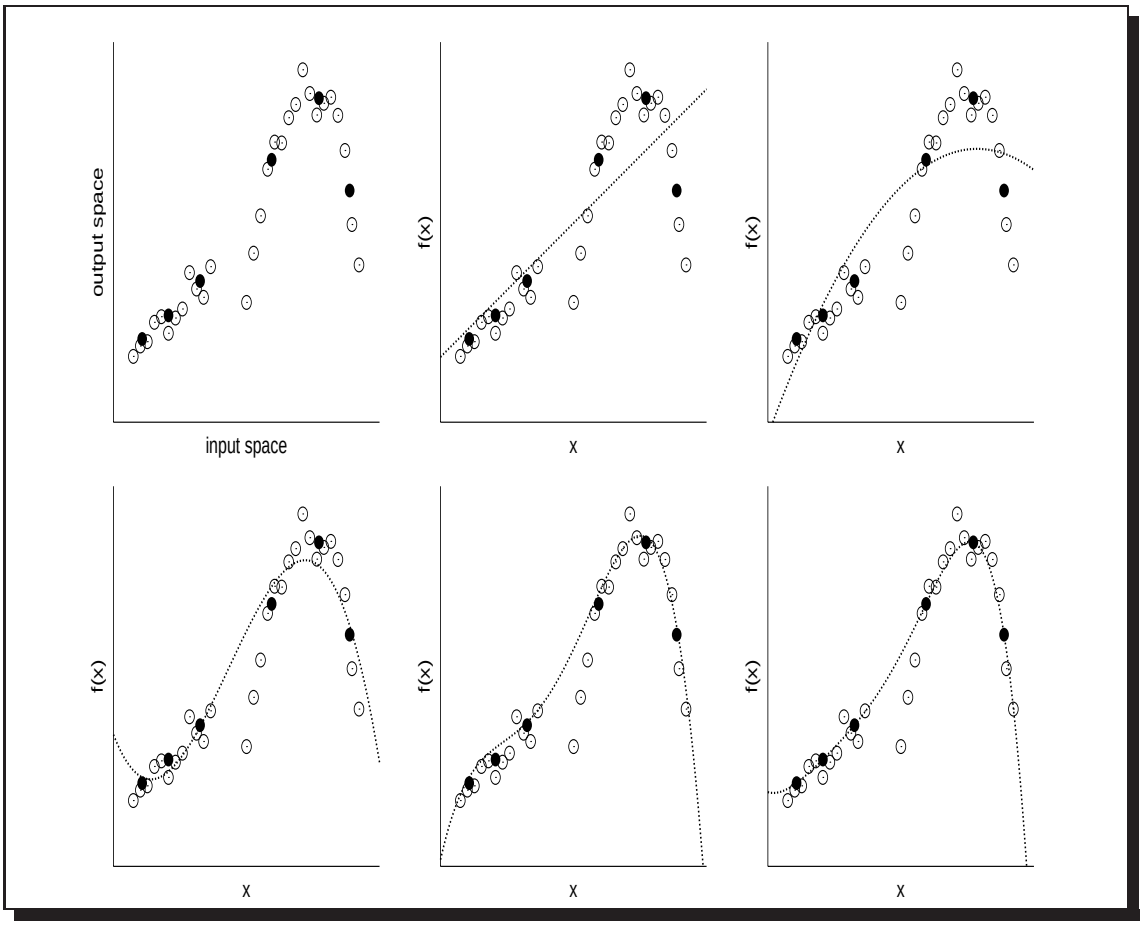

Figure 1.1. An illustration of the classification problem.

function approximation process, in which multiple mapping functions are built. Each mapping function is likely to perform particularly accurate predictions for the inputs that define the corresponding sub-problem.

This finer-grained process is illustrated in Figure 1.2. The original problem is decomposed into two sub-problems. One sub-problem is defined by the three examples (i.e., first three black points), while the other sub-problem is defined by the last three examples. Two mapping functions are built using the respective set of examples. Each mapping function provides an optimized approximation of the target function on specific regions of the input space. Although this strategy is very intuitive, some key questions must be answered:

- How sub-problems are defined/differentiated?
- Is there a suitable way to search for mapping functions, such that the space for candidate functions is constrained?
- Can particularities of a sub-problem be used to improve function approximation?

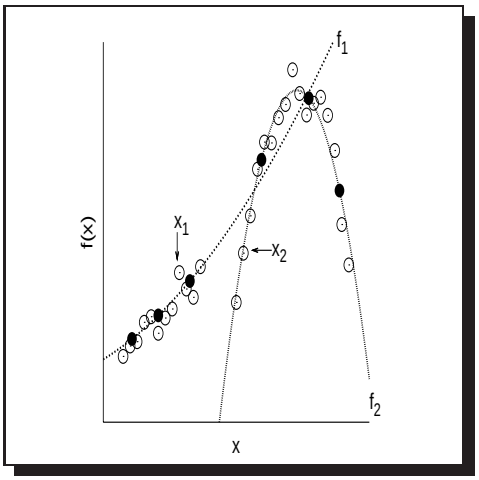

Figure 1.2. Decomposition into sub-problems.

- Is the approximation obtained by multiple functions tighter than the approximation obtained by a single function?

- What is the computational cost associated with algorithms that use multiple mapping functions to approximate the target function? Are there polynomial time, efficient algorithms? Are they more efficient than algorithms that approximate the target function using a single function?

This thesis is mainly devoted to answer these questions.

1.1 Thesis Statement

Classification is posed as a function approximation problem, which can be decomposed into sub-problems that are defined by different regions of the input space. The main hypothesis of this thesis is that such sub-problems are much simpler than the original problem. The aim of this thesis is to show that, instead of of approximating the target function in the full space of inputs, approximating the target function in specific ranges or regions of the input space, on a demand-driven basis (i.e., taking into account particular demands of each region of the input space) leads to more accurate mapping functions.

1.2 Thesis Contributions

Some of the specific contributions of this thesis include:

- Associations between inputs and outputs are discovered using a well-known data mining technique. These associations are used to constrain the space for mapping functions to only those functions that are likely to be accurate. We show, in Chapter 3, that this strategy leads to algorithms that need few examples to build accurate mapping functions (i.e., Probably Approximately Correct, or simply, PAC-efficient algorithms).

- We show, in Chapter 4, that different sub-problems demand different mapping functions.

- We propose polynomial-time algorithms for demand-driven associative classification in Chapter 4.

- Several extensions to demand-driven associative classification are presented in Chapter 5.

- An extensive set of experiments demonstrates the effectiveness of the proposed algorithms in various scenarios and applications.

1.3 Informal Description

The goal of this informal description is to help you, the reader, understand what this thesis is about, and what you will learn if you chose to read it. We will attempt to give a picture not only of the research itself, but of the choices and developments that led to this research.

To a large extent, this thesis is about Associative Classification and related algorithms. They are machine learning algorithms for solving classification problems. The core of these algorithms is the explicit use of association rules (which is a typical data mining technique) expressing the relationship between features and classes. When we first began the research that led to this thesis, in 2005, there was a growing sense that associative classification algorithms were not of practical use in complex applications. This was mainly due to the exponential number of rules that could be extracted from the training data.

At that time, we already had some experience developing efficient association rule mining algorithms [Veloso et al., 2002a,b, 2003, 2002c], and we felt we could, somehow, solve this impediment, and make associative classification algorithms more efficient and practical. Our insight was that, since classification performance is usually assessed using a test set, then to achieve high classification performance, only a much much smaller subset of rules needs to be extracted, more specifically, only those rules

that carry discriminative information about instances in the test set. This observation motivated our first paper in this field: Lazy Associative Classification (abbreviated as LAC) which introduced lazy algorithms for associative classification and was published in [Veloso et al., 2006b]. LAC independently solves sub-problems by "projecting" the training data according to instances in the test set, so that only rules that contribute somehow to improve classification performance are processed. Two months later, we improved LAC and applied it to solve complex classification problems, such as document categorization. These algorithms were able to achieve pretty good results, which were published in [Veloso et al., 2006a].

Then, we started to apply LAC to all problems we had some interest in solving. This included spam filtering, protein functional analysis, social networks, sentiment analysis, and many others. Most of these attempts to solve interesting problems resulted in publications [Veloso and Meira, 2006, 2005, 2007; Veloso et al., 2007b].

We then turned to the analysis of variations of the original classification problem. We extended LAC so that it became able to solve multi-label problems. Our first attempt was, however, over-simplified, and the results we obtained were not as promising as we were expecting. Fortunately, we observed that, frequently, different labels are somehow associated to each other. Thus, we decided to exploit the association between labels in order to improve our algorithms. We were able to solve several multi-label problems with these algorithms, which were published in [Veloso et al., 2007a]. Another task that is related to classification is ranking. Putting simple, the major difference is that, instead of learning how to group objects, one has to learn how to sort them. We were able to extend LAC so that it became able to solve ranking problems. In fact, LAC is currently one of the most effective algorithms that learn ranking functions, in the context of information retrieval. Exciting results were published in [Veloso et al., 2008a].

During the process of implementing and reimplementing LAC, we realized that each projection is, in fact, a sub-problem which could be solved using specifically designed strategies. Some sub-problems are extremely simple, while others are very hard to solve. Such a finer-grained approach may combine simple and complex solutions in order to solve the original problem. For example, selecting which statistic measure of association is the best for each sub-problem leads to overall improvements in classification performance, as was shown in [Veloso et al., 2009c].

From now on, this thesis will be written in a more technical, formal style.

1.4 Thesis Outline

This thesis is structured in 6 chapters. The remainder of this thesis is organized as follows.

Chapter 2. [**The Classification Problem**] Basic definitions, notations, challenges and techniques concerning the classification problem are presented.

Chapter 3. [**Associative Classification**] Algorithms that produce candidate functions by exploiting associations between inputs and outputs are presented. These algorithms are denoted as associative classification algorithms.

Chapter 4. [**Demand-Driven Associative Classification**] Algorithms that use multiple functions to approximate the target function are presented. These algorithms are denoted as demand-driven associative classification algorithms. Empirical results showing the effectiveness of these algorithms are reported.

Chapter 5. [**Extensions to Demand-Driven Associative Classification**] Several extensions to demand-driven associative classification are discussed.

Chapter 6. [**Conclusions**] Contributions and limitations are summarized and the thesis is concluded.

Chapter 2

The Classification Problem

In this chapter we describe basic definitions that are necessary to understand the classification problem. Further, we also discuss some of the main challenges and research wreathing this problem.

2.1 Definitions

In this section we present definitions and notations that form the basis of the classification problem.

Training Data and Test Set

In a classification problem, there is a set of input-output pairs (also referred to as instances or examples) of the form z_i=(x_i, y_i). Each input x_i is a fixed-length record of the form $< a_1, \ldots, a_l >$, where each a_i is an attribute-value. Each output y_i draws its value from a discrete and finite set of possibilities $y = \{c_1, \ldots, c_p\}$, and indicates the class to which z_i belongs. Cases where y_i =? indicate that the correct class of z_i is unknown. There is a fixed but unknown conditional probability distribution P$(y|x)$, that is, the relationship between inputs and outputs is fixed but unknown. The set of pairs is explicitly divided into two partitions, the training data (denoted as $\mathcal{S}$) and the test set (denoted as $\mathcal{T}$):

$$\begin{aligned} \mathcal{S} &= \{s_1 = (x_1, y_1), \ldots, s_n = (x_n, y_n)\} \\ \mathcal{T} &= \{t_1 = (x_{n+1}, ?), \ldots, t_m = (x_{n+m}, ?)\} \end{aligned}$$

Further, it is assumed that pairs in $\mathcal{T}$ are in some sense related to pairs in $\mathcal{S}$, and that $\{t_{n+1}, \ldots, t_{n+m}\}$ and $\{s_1, \ldots, s_n\}$ are sampled independently and identically from

the same distribution $P(y|x)$.

Classification Algorithm

A classification algorithm takes as input the training data $\mathcal{S}$ and the test set $\mathcal{T}$, and returns a mapping function $f_{\mathcal{S}} : x \rightarrow y$ that represents the relation between inputs and outputs in $\mathcal{S}$, that is, the mapping function $f_{\mathcal{S}}$ is a discrete approximation of $P(y|x)$ (i.e., a classification algorithm observes n input-output pairs and produces a function which describes well the underlying input-output process). Many possible functions can be derived from $\mathcal{S}$. The hypothesis space $\mathcal{H}$ is the space of functions explored by the classification algorithm in order to select $f_{\mathcal{S}}$. The selected mapping function $f_{\mathcal{S}}$ is finally used to estimate the outputs y given the inputs x, for each $x_i \in \mathcal{T}$.

Figure 2.1 illustrates the problem of function approximation. The dark solid line represents the true (target) function. The dark points are given as examples (i.e., $\mathcal{S} = \{s_1, \ldots, s_n\}$). Two approximations (i.e., candidate functions) are used to fit the true function. The complex approximation fits $\mathcal{S}$ exactly. Yet, it is clear that the complex approximation will perform poorly in $\mathcal{T}$, as it is far from the true function on most of the space of inputs (i.e., the x-axis). The simple approximation does not fit $\mathcal{S}$ exactly, but provides better approximations for most of the points in $\mathcal{T}$. The classification problem is that of selecting, from all functions in $\mathcal{H}$, the one which best approximates (discretely) the distribution $P(y|x)$. The selection is based on $\mathcal{S}$. This formulation implies that the classification problem corresponds to the problem of function approximation.

Loss Function

A loss function, $\ell(f_{\mathcal{S}}, z_i = (x_i, y_i))$, represents the loss (or cost) associated with a wrong estimate (i.e., $f_{\mathcal{S}}(x_i) \neq y_i$) as a function of the degree of deviation from the correct value. Unless otherwise stated, the 0-1 loss function will be the one used throughout this thesis, where for z_i=(x_i, y_i):

$$\ell(f_{\mathcal{S}}, z_i) = \begin{cases} 0 & \text{if } f_{\mathcal{S}}(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$

The 0-1 loss function is very intuitive, since it states that one should make as few mistakes as possible. It may be considered an upper bound for other loss functions, such as the hinge and the squared loss functions [Rosasco et al., 2004].

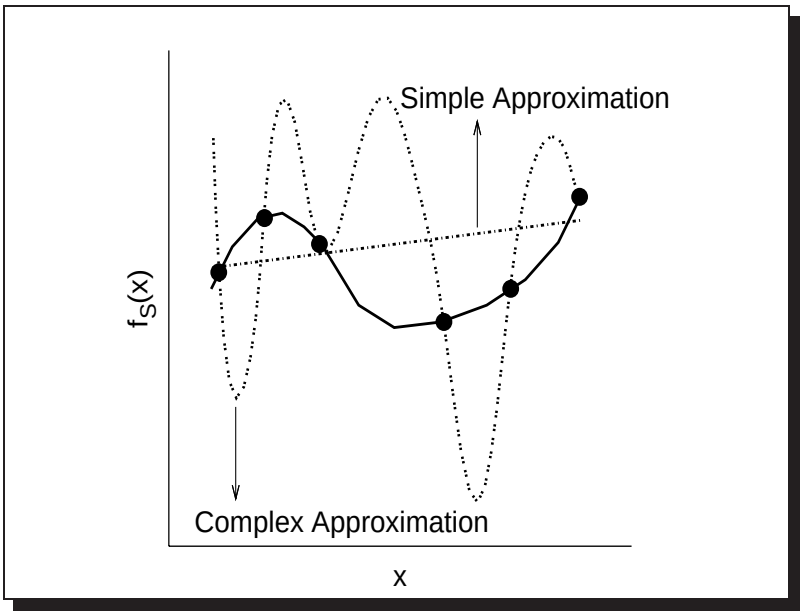

Figure 2.1. Simple and complex mapping functions.

Expected Error

According to Cucker and Smale [2001] and Vapnik [1995], the expected error of a mapping function $f_{\mathcal{S}}$ is defined as:

$$\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] = \int_{t=(x,y)} \ell(f_{\mathcal{S}}, t)\mathrm{dP}(y|x)$$

The primary goal of classification algorithms is to select a mapping function $f_{\mathcal{S}}$ for which $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$ is guaranteed low. However, $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$ cannot be computed because the conditional probability distribution $\mathrm{P}(y|x)$ is unknown.

Empirical Error

Although the expected error is unknown, the empirical error of a mapping function $f_{\mathcal{S}}$ can be easily computed using $\mathcal{S}$:

$$\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\mathcal{S}}, s_i)$$

Generalization

An important ability for any classification algorithm is generalization: the empirical error must converge to the expected error as the number of examples n increases, that is, $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] \approx \mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$. Informally, the classification performance of the selected function,

$f_{\mathcal{S}}$, in $\mathcal{S}$ must be a good indicator of its classification performance in $\mathcal{T}$. Generalization error (or risk), denoted as ϵ, is given by $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] - \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}]$. High generalization (i.e., low values of ϵ) implies low expected error only if $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] \approx 0$.

Next we discuss a well-known mathematical tool for the analysis of classification algorithms.

2.2 The Probably-Approximately Correct Learning Framework

The Probably-Approximately Correct (PAC) learning framework [Valiant, 1984a,b] states that the classification algorithm must be able to select a mapping function $f_{\mathcal{S}}$ from $\mathcal{H}$ which, with high probability, will have low expected error. There are two major requirements in the PAC learning framework:

- The expected error is bounded by some constant ϵ (i.e., the generalization error).
- The probability that the expected error is greater than ϵ is bounded by some constant δ.

Putting simple, the PAC learning framework requires that the classification algorithm *probably* selects a mapping function $f_{\mathcal{S}}$ that is *approximately correct*. More specifically, a classification problem is PAC-feasible if the algorithm selects a mapping function $f_{\mathcal{S}} \in \mathcal{H}$, such that $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] \leq \epsilon$, with probability of at least $(1-\delta)$, for $0 < \epsilon < \frac{1}{2}$ and $0 < \delta < \frac{1}{2}$. This statement is formalized as follows:

$$\mathrm{P}[\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] < \epsilon] \geq 1 - \delta \tag{2.1}$$

Sample Complexity

The sample complexity of a classification algorithm is the relation between $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$ and $|\mathcal{S}|$ (or n). Inequality 2.1 can be used to derive the sample complexity of a classification algorithm. In this case, a mapping function, $f_{\mathcal{S}}$, is considered accurate if $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] < \epsilon$. We denote an accurate function as f^+, and similarly, we denote poor functions as f^-. Also, f^* is the most accurate mapping function in the hypothesis space, $\mathcal{H}$.

For a given pair $z_i = (x_i, y_i)$ (i.e., an example), the probability of $f^-(x_i) \neq y_i$, is at least ϵ. Thus, the probability of $f^-(x_i) = y_i$ is at most $1 - \epsilon$. So, for n pairs $\{z_1 = (x_1, y_1), \ldots, z_n = (x_n, y_n)\}$, the probability that $f^-(x_1) = y_1 \wedge \ldots \wedge f^-(x_n) = y_n$ is at most $(1-\epsilon)^n$. Now, considering that there are k poor functions in $\mathcal{H}$, the probability that at least one of these functions correctly predicts the output of the n pairs is

$k \times (1-\epsilon)^n$. Using the fact that $k \leq |\mathcal{H}|$ (and assuming that $\mathcal{I}_{\mathcal{T}}[f^*] = 0$), the following inequality is obtained:

$$\mathrm{P}[\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] > \epsilon] \leq \mathcal{H} \times (1-\epsilon)^n \leq \delta \tag{2.2}$$

Since $(1-\epsilon) \leq e^{-\epsilon}$ [Kearns and Vazirani, 1994], and solving for n, (2.2) can be rewritten as:

$$\begin{aligned} \mathrm{P}[\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] > \epsilon] \leq \mathcal{H} \times e^{-n\epsilon} &\leq \delta \\ \mathcal{H} \times e^{-n\epsilon} &\leq \delta \\ n &\geq \frac{1}{\epsilon}\big(\ln|\mathcal{H}| + \ln(\frac{1}{\delta})\big) \end{aligned} \tag{2.3}$$

Thus, the more accuracy (lower ϵ values) and the more certainty (lower δ values) one wants, the more examples the classification algorithm needs. Now, (2.2) and (2.3) can be used to derive the expected error bound:

$$\begin{aligned} \epsilon &\geq \frac{1}{n}\big(\ln|\mathcal{H}| + \ln(\frac{1}{\delta})\big) \\ \mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] &\leq \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] + \frac{1}{n}\big(\ln|\mathcal{H}| + \ln(\frac{1}{\delta})\big) \end{aligned} \tag{2.4}$$

So far, it was assumed that $\mathcal{I}_{\mathcal{S}}[f^*] = 0$ (i.e., the classification algorithm is gnostic[1]). If $\mathcal{I}_{\mathcal{S}}[f^*] > 0$ (i.e., the classification algorithm is agnostic), then, according to Angluin [1992], Chernoff approximation can be used to derive the sample complexity:

$$n \geq \frac{1}{2\epsilon^2}\big(\ln|\mathcal{H}| + \ln(\frac{1}{\delta})\big) \tag{2.5}$$

Now, (2.2) and (2.5) can be used to derive the expected error bound:

$$\begin{aligned} \epsilon &\geq \sqrt{\frac{1}{2n}\big(\ln|\mathcal{H}| + \ln(\frac{1}{\delta})\big)} \\ \mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] &\leq \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] + \sqrt{\frac{1}{2n}(\ln|\mathcal{H}| + \ln(\frac{1}{\delta}))} \end{aligned} \tag{2.6}$$

[1] A function $f_{\mathcal{S}}$ is consistent with example $s = (x, y)$ if $f_{\mathcal{S}}(x) = y$. A classification algorithm is gnostic if it selects a function $f_{\mathcal{S}}$ which is consistent with all examples in $\mathcal{S}$.

For PAC-based expected error bounds, $|\mathcal{H}|$ must be estimated. The simpler the hypothesis space (or, equivalently, the fewer functions are explored), the lower is ϵ, at the expense of increasing the empirical error.

Classification Efficiency

The empirical error is a finite sample approximation of the expected error. It can be shown [Cucker and Smale, 2001] that the empirical error converges uniformly to the expected error when $|\mathcal{S}| \to \infty$ ($n \to \infty$). An efficient classification algorithm ensures that this convergence occurs with high rate. Formally, in the PAC learning framework, a classification algorithm is efficient if it selects, in polynomial time and with a polynomial number of examples, with probability $(1-\delta)$, a function $f_{\mathcal{S}} \in \mathcal{H}$ for which $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] < \epsilon$, and $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] \approx \mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$ (that is, efficient classification algorithms must achieve low empirical error, with access to a restricted number of examples and in a reasonable amount of time).

2.3 Function Approximation

Classification is posed as synthesizing a mapping function that best approximates the relationship between the inputs x_i and the corresponding outputs y_i (i.e., the classes). Two strategies for function approximation are considered in this thesis: empirical risk minimization (which follows the stability theory [Devroye and Wagner, 1979; Kutin and Niyogi, 2002; Bousquet and Elisseeff, 2002; Mukherjee et al., 2006]), and structural risk minimization (which follows the VC theory [Guyon et al., 1992; Vapnik, 1991, 1995]). Both strategies establish sufficient conditions for generalization. Next we will discuss these strategies.

Empirical Risk Minimization

Probably the most natural function approximation strategy is Empirical Risk Minimization (ERM): from all possible mapping functions in $\mathcal{H}$, the classification algorithm selects the function $f_{\mathcal{S}}$ which minimizes $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}]$, the empirical error given by:

$$\arg\min \left(\frac{1}{n} \sum_{i=1}^{n} \ell(f_{\mathcal{S}}, s_i) \right), \forall f_{\mathcal{S}} \in \mathcal{H} \tag{2.7}$$

The Empirical Risk Minimization strategy, however, does not ensure generalization. More specifically, minimizing the empirical error does not necessarily imply in mini-

mizing the expected error. A sufficient condition for generalization of ERM algorithms is the stability of $f_{\mathcal{S}}$ [Mukherjee et al., 2006; Poggio et al., 2004].

Stability. The stability measures the difference, β_{s_i}, in empirical errors at a pair $s_i \in \mathcal{S}$ between a function $f_{\mathcal{S}}$ obtained given the entire training data $\mathcal{S}$ and a function $f_{\mathcal{S}-s_i}$ obtained given the same training data but with pair s_i left out. Specifically, if the training data $\mathcal{S}$ is perturbed by removing one pair s_i, and if the selected function $f_{\mathcal{S}}$ does not diverge much from $f_{\mathcal{S}-s_i}$, then $f_{\mathcal{S}}$ is stable. Informally, avoiding unstable functions can be thought as a way of controlling the variance of the function approximation process. Function $f_{\mathcal{S}}$ is β-stable if:

$$\forall s_i \in \mathcal{S}, |f_{\mathcal{S}}(s_i) - f_{\mathcal{S}-s_i}(s_i)| \leq \beta \tag{2.8}$$

The lowest value of β in (2.8) provides the stability of $f_{\mathcal{S}}$. The lowest value of β is the largest change at any pair s_i. Thus, function $f_{\mathcal{S}}$ shown in Figure 2.2, is obtained by Empirical Risk Minimization using $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5\}$. Similarly, function $f_{\mathcal{S}-s_2}$ is obtained by Empirical Risk Minimization using $\{\mathcal{S}-s_2\}$, and function $f_{\mathcal{S}-s_5}$ is obtained by Empirical Risk Minimization using $\{\mathcal{S} - s_5\}$. The difference at s_2, β_{s_2}, is small. The difference at s_5, β_{s_5}, is large. Therefore, $f_{\mathcal{S}}$ is β_{s_5}-stable, despite the very small value of β_{s_2}. Function $f_{\mathcal{S}}$ is stable if $\beta = O(\frac{1}{n})$.

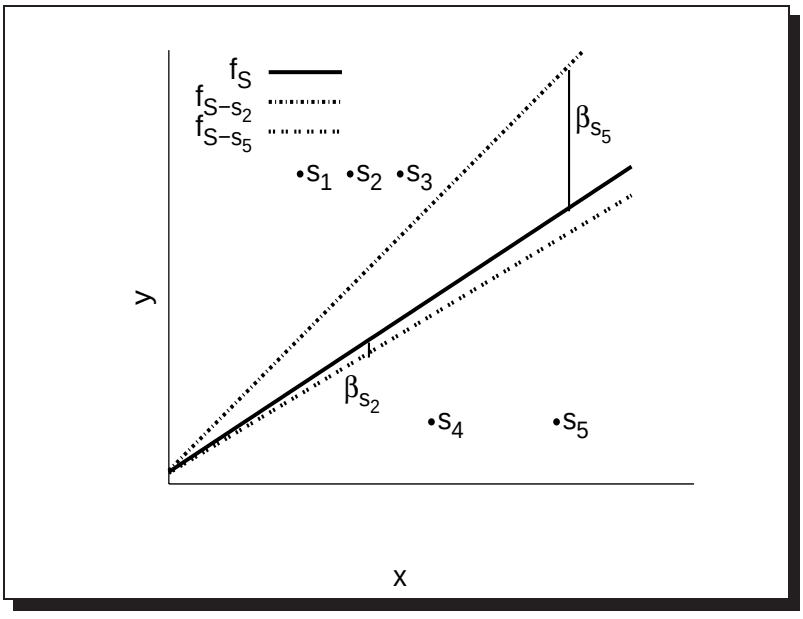

Figure 2.2. Empirical risk minimization.

It has been shown in [Bousquet and Elisseeff, 2002] that the expected error can be estimated by the empirical error and the stability of the selected function $f_{\mathcal{S}}$, as follows:

$$\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] \leq \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] + \left(\beta + (4n\beta + 1) \times \sqrt{\frac{\ln \frac{1}{\delta}}{2n}} \right) \tag{2.9}$$

Thus, the function $f_{\mathcal{S}}$ which minimizes $\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$ can be selected by applying (2.9) to each possible candidate function.

Structural Risk Minimization

Structural Risk Minimization (SRM) provides a trade-off between the complexity of a function and its empirical error. Simpler functions may provide high empirical error (they may underfit the training data), while complex functions may provide low empirical error (but it may be by means of overfitting the training data). Thus, from all possible functions in $\mathcal{H}$, the classification algorithm following the Structural Risk Minimization strategy selects the function $f_{\mathcal{S}}$ which minimizes this trade-off.

A structure is a (possibly infinite) nested set of classes of functions $\mathcal{F}_i$, such that $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots$, where functions in $\mathcal{F}_1$ are simpler (i.e., have lower complexity) then functions in $\mathcal{F}_2 - \mathcal{F}_1$, and so on. Since classes of functions are nested, the empirical error decreases as the complexity of classes of functions increases.

The Vapnik-Chervonenkis Dimension – Suppose $y_i \in \{\pm 1\}$ (i.e., examples in $\mathcal{S}$ are classified either as positive or as negative). In this case, n pairs in $\mathcal{S}$ can be labeled in 2^n ways as positive and negative. Therefore, 2^n different classification problems can be defined by n pairs. If for any of these problems, the function $f_{\mathcal{S}} \in \mathcal{H}$ exactly separates all the positive examples from all the negative ones, then it is said that $f_{\mathcal{S}}$ shatters n pairs (i.e., all pairs in $f_{\mathcal{S}}$ can be classified with no error by $f_{\mathcal{S}}$). The maximum number of pairs in $\mathcal{S}$ that can be shatttered by $f_{\mathcal{S}}$ is called the VC-dimension [Vapnik and Chervonenkis, 1971; Blumer et al., 1989] of $f_{\mathcal{S}}$, which is denoted as $d_{f_{\mathcal{S}}}$.

An example of classification problem composed of four pairs in two dimensions (i.e., attributes a_1 and a_2) is given in Figure 2.3. A rectangle can shatter four points in two dimensions, but a line can shatter only three. Thus, for this classification problem, the VC-dimension of a rectangle is four, while the VC-dimension of a line is three. For a given classification problem, $d_{f_{\mathcal{S}}}$ is given by the maximum number of pairs that can be correctly classified by $f_{\mathcal{S}}$. VC dimension may seem pessimistic, since it establishes that a line can only classify problems composed of three pairs, and not more. A function that can classify only three pairs is not very useful. However, this is because the VC dimension is independent of the probability distribution from which pairs are drawn

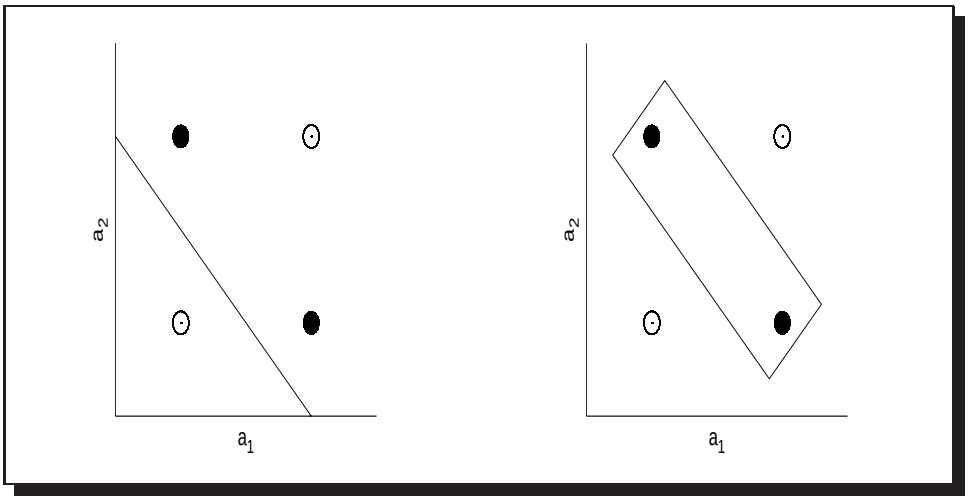

Figure 2.3. The VC-dimension.

(i.e., P($y|x$)). In practice, however, pairs that are close to each other often have the same label.

Complexity. The VC-dimension measures the expressive power, richness or flexibility of a set of functions by assessing how wiggly its members can be. The complexity of a function $f_{\mathcal{S}}$ is usually given by the VC-dimension of $f_{\mathcal{S}}$. It has been shown [Guyon et al., 1992] that the expected error can be estimated by the empirical error and the complexity of the selected function $f_{\mathcal{S}}$, as follows:

$$\mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}] \leq \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] + \sqrt{\frac{d_{f_{\mathcal{S}}}\left(\ln\frac{2n}{d_{f_{\mathcal{S}}}}+1\right) - \ln\frac{\delta}{4}}{n}} \tag{2.10}$$

The shape of this bound, which is shown in Figure 2.4, can be exploited to select a function with the most appropriate complexity for $\mathcal{S}$. Classes of functions are considered in increasing complexity (i.e., $\mathcal{F}_1$ is considered before $\mathcal{F}_2$, and so on). Structural Risk Minimization corresponds to finding the simplest function $f_{\mathcal{S}}$ which provides the lowest empirical error. By applying (2.10) to each class of functions, a function in the class for which the error bound is tightest can be selected.

2.4 Major Challenges

In this section some of the current challenges regarding the problem are outlined. This list is not exhaustive and is intended to give the reader a feel of the types of challenge we wrestle with.

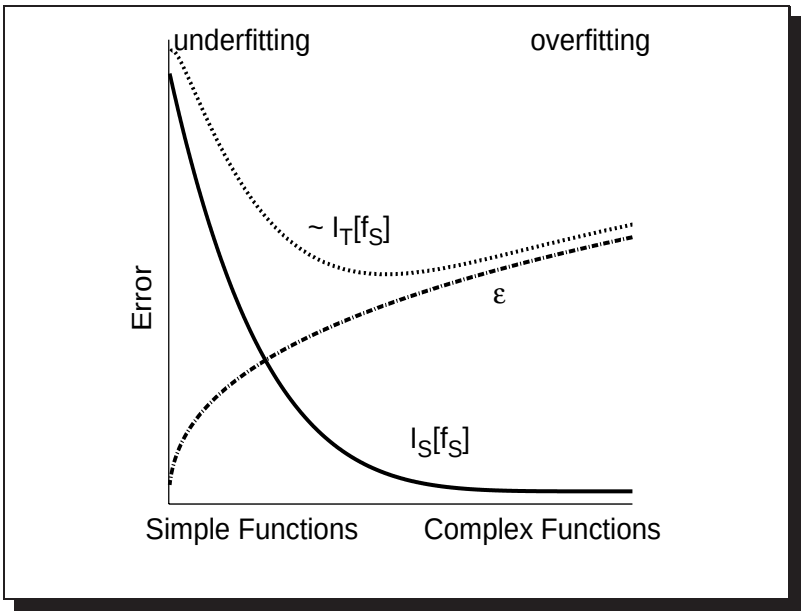

Figure 2.4. Structural risk minimization.

High Dimensionality

The hypothesis space, $\mathcal{H}$, increases exponentially with the number of attributes in $\mathcal{S}$ (i.e., the number of dimensions). Thus, as the dimensionality of $\mathcal{S}$ increases, it becomes exponentially more difficult to find the best function in $\mathcal{H}$. This means that functions (i.e., classification problems) that are specified by many attributes are usually hard to approximate[2]. A simple, and sometimes very effective way of dealing with high-dimensional problems is to reduce the number of attributes by eliminating those that seem irrelevant. This suggests that classification algorithms must be able to eliminate the largest amount of irrelevant attributes without discarding any important information, so that the number of candidate functions is reduced.

Labeling

Annotating (or labeling) data usually requires humans who can explicitly provide examples showing the relationship between inputs and outputs. Thus, the acquisition of labeled training examples may be very costly, time-consuming, and prone to error. Labeling a single example for protein shape classification, for instance, can take months.

[2]Suppose a classification problem in which each x_i is composed of l binary attributes, and $y_i \in \{\pm 1\}$. In this case, there are 2^l possible input patterns and 2^{2^l} possible functions that describe the relationship between x_i and y_i. Given n examples in $\mathcal{S}$, there will be $\frac{2^{2^l}}{2^n} = 2^{(2^l - n)}$ possible functions that can be extracted from $\mathcal{S}$. If $l = 10$ and $n = 1,000$, then the number of possible functions will be $2^{(1,024-1,000)} \approx 4$ million possible functions. Thus, although $\mathcal{S}$ contains almost all possible examples, the number of possible functions is still enormous.

Documents, videos and images on the Internet may contain very ambiguous content, challenging the expertise of the annotators. This suggests that classification algorithms must be able to somehow exploit unlabeled data, in order to reduce the annotation burden.

Outliers

An outlier is an instance that is numerically distant from most of the examples in the training data. Classification algorithms, specially those that follow the ERM function approximation strategy may erroneously select complex functions in order to fit outliers. Thus, outliers can be extremely harmful to effective classification.

Processing Time

Processing time has been an influential factor in designing classification algorithms. Some algorithms are extremely slow in learning classification functions (i.e., training time). Others are fast in learning classification functions, at the price of slowing down the assignment of classes to test instances (i.e., classification time). Efforts to reduce processing time have been mainly pursued by significantly reducing the number of attributes without sacrificing necessary information. This suggests that classification algorithms must be able to discard only useless attributes.

Next we discuss some widely used classification methods.

2.5 Classification Methods

There has been a huge proliferation of classification methods. In the following, some of the most popular methods (and which we believe are representative of the state-of-the-art), will be briefly described. The goal is not to go too much into detail, but to show the basic principles and the differences between these methods.

2.5.1 Decision Trees (DTs)

The decision tree classification method expresses the relationship between inputs x_i and outputs y_i using a tree. Each interior node of the tree corresponds to an attribute, and a connection from this node to one of its children represents a possible value of the corresponding attribute. A leaf node represents the predicted output (or class), given the values of the attributes represented by the path from the root. Thus, a path is essentially conjunctions of attribute-values that lead to a prediction.

Top down algorithms for building decision trees usually follow a divide-and-conquer recursive strategy [Quinlan, 1986, 1993], which splits a node into its children. This process is repeated on each derived child in a recursive manner. The recursion is completed when splitting is either non-feasible, or perfect discrimination is reached. The crucial step is the attribute selection (i.e., which attribute must be included at a given node). Several attribute selection criteria were already proposed, such as entropy-based selection measures, which will be detailed next.

Entropy-Based Selection Let V be a discrete random variable with range $\mathcal{V}$. In this case, the entropy of V (also known as the information of V), is defined in Equation 2.11, where $0 \log 0 = 0$ and the base of the logarithm is two, so that entropy is expressed in bits. The entropy is always non-negative and quantifies the amount of uncertainty of V.

$$E(V) = -\sum_{v \in \mathcal{V}} p(V = v) \log p(V = v) \tag{2.11}$$

The conditional entropy of V given another random variable, Q (with range $\mathcal{Q}$), is the expected value of the entropies of the conditional distributions averaged over the conditioning random variable, which is given in Equation 2.12.

$$\begin{aligned} E(V|Q) &= -\sum_{q \in \mathcal{Q}} E(V|Q = q) \\ &= -\sum_{q \in \mathcal{Q}} p(Q = q) \sum_{v \in \mathcal{V}} p(V = v|Q = q) \log p(V = v|Q = q) \\ &= -\sum_{q \in \mathcal{Q}} \sum_{v \in \mathcal{V}} p(V = v \wedge Q = q) \log p(V = v|Q = q) \end{aligned} \tag{2.12}$$

The mutual information of V and Q, also known as the *information gain* of V given Q ($\mathcal{G}(V;Q)$), quantifies the relative entropy between the joint distribution and the product distribution, as shown in Equation 2.13.

$$\begin{aligned} \mathcal{G}(V;Q) &= \sum_{v \in \mathcal{V}} \sum_{q \in \mathcal{Q}} p(V = v \wedge Q = q) \log \frac{p(V = v \wedge Q = q)}{p(V = v)p(Q = q)} \\ &= E(V) - E(V|Q) \end{aligned} \tag{2.13}$$

Information gain quantifies the reduction in uncertainty in V after observing the value of Q. Given the training data, the information gain can be computed by using the

empirical probabilities, with V representing the outputs (or classes) and Q representing the possible values of an attribute. The attribute selection step is implemented by testing the information gain for each attribute Q with outputs V, and picking the attribute associated with the highest information gain, as illustrated in Figure 2.5. In the figure, splitting process is illustrated from left to right. First, splitting the y-axis provides the highest information gain. Then, the x-axis is also splitted. In the end of the process, two partitions of the space are induced by the decision tree.

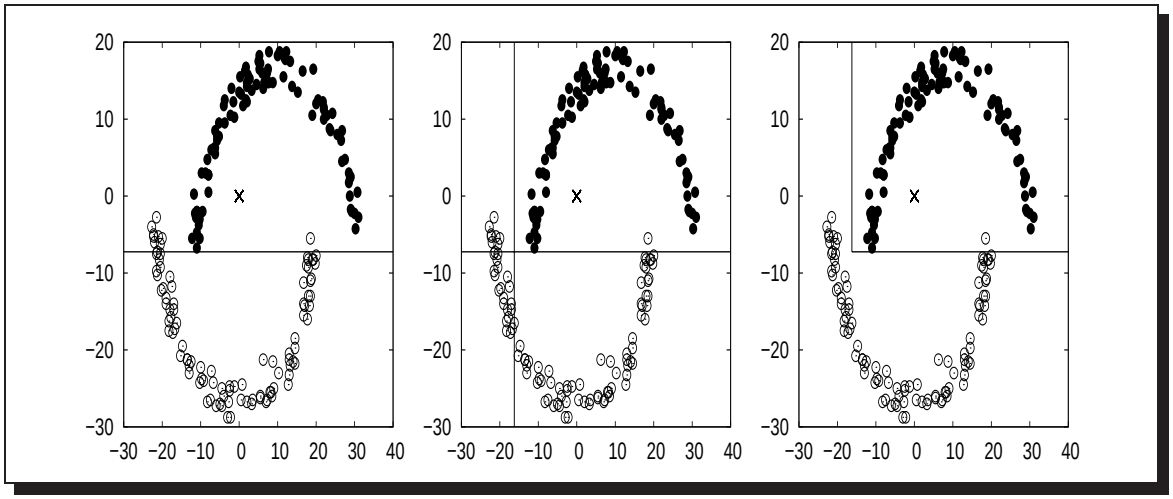

Figure 2.5. Splitting according to information gain.

2.5.2 Naive Bayes (NB)

The Naive Bayes classification method is commonly studied in machine learning [Domingos and Pazzani, 1997; Wolpert, 1995]. The basic idea is to use the joint probabilities of attribute-values and outputs (or classes) to estimate the probabilities of each output given a test instance. The naive part is the assumption of independence between attributes, that is, the conditional probability of an attribute-value a_i given an output y is assumed to be independent from the conditional probabilities of another attribute-value a_j given that output:

$$p(a_i|y \wedge a_j) = p(a_i|y)$$

This assumption makes NB algorithms usually very fast, because it does not use combinations of attribute-values as predictors. In spite of their naive design and apparently over-simplified assumptions, NB algorithms often perform much better than it might be expected [Zhang, 2005].

2.5.3 Nearest Neighbors (NNs)

The k-nearest neighbors classification method has been extensively studied [Dasarathy, 1990]. Given an input $x_i \in \mathcal{T}$, the classification is based on the k closest training inputs in the attribute space. The space is partitioned into regions (or locations) depending on x_i and k, as shown in Figure 2.6. In the figure, the number of neighbors increases from left to right. Usually, Euclidean distance is used in order to place the examples (or points) in the proper location in the space.

k-NN is a lazy classification method, in which the training phase consists only of storing the examples and the corresponding classes. During testing phase, distances are computed and the k closest inputs to x_i are selected. There is a number of ways to predict the output for x_i. Typically, the predicted output for x_i is the most common class amongst the k nearest neighbors.

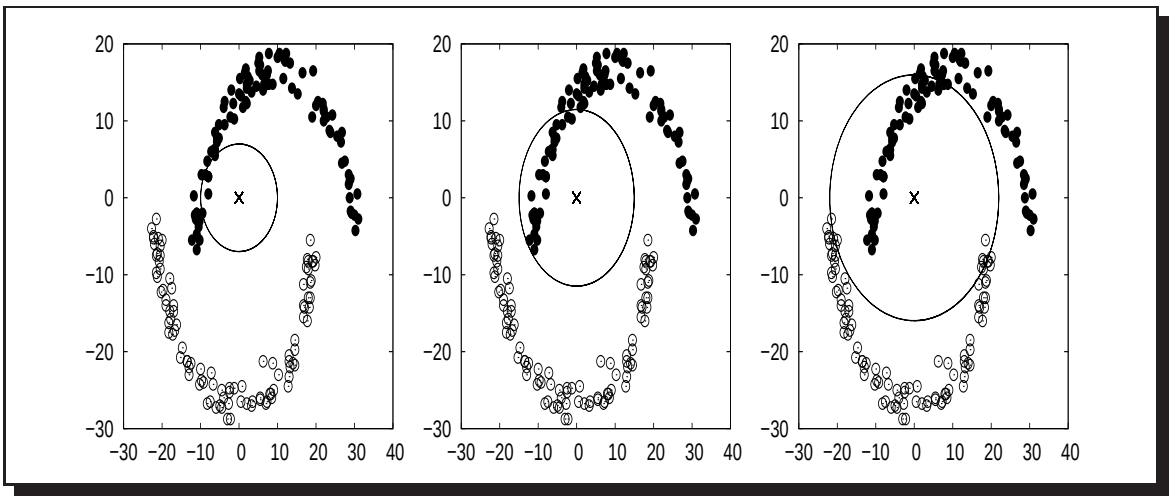

Figure 2.6. Increasing the number of neighbors.

2.5.4 Support Vector Machines (SVMs)

The Support Vector Machine classification method was introduced in [Boser et al., 1992]. The method was initially developed for solving problems with only two classes (i.e., there are only two possible outputs), but it can be extended to solve more complex problems. SVMs are defined over the attribute space, and the problem is to find a decision surface that best separates inputs that are related to different outputs.

Margin In order to define the best separation, it is necessary to introduce the definition of *margin* between two outputs (or classes). Figure 2.7 illustrates the idea. For simplicity, it is shown only a case in a two-dimensional space with linearly separable

classes, but the idea can be generalized to a high dimensional space and to classes that are not linearly separable. A decision surface in a linearly separable space is a hyperplane. The solid lines in Figure 2.7 show two possible decision surfaces, each of which correctly separates the two groups of instances. The dashed lines parallel to the solid ones show how much one can move the decision surface without causing errors. The distance between each set of those parallel lines are referred to as the margin. Graph on the left shows a decision line (solid) with a smaller margin, while graph on the right shows the decision line (solid) with maximal margin. SVMs find the decision surface that maximizes the margin between the instances in the training data. The assumption is that the larger the margin the higher the generalization will be.

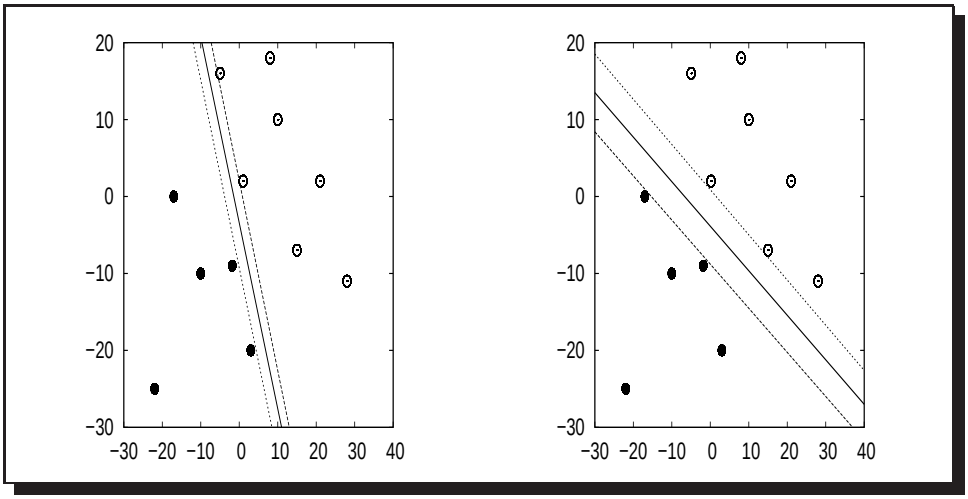

Figure 2.7. Maximum margin hyperplane.

More precisely, this decision surface is a hyperplane which is written in Equation 2.14, where x_i is an arbitrary test instance, and w and the constant b are learned from the training data.

$$w \cdot x_i - b = 0 \tag{2.14}$$

Let $y_i \in \{\pm 1\}$ be the possible classes for x_i (i.e., assuming only two classes). The SVM problem is to find w and b that satisfy the constraints shown in Equations 2.15 and 2.16, which can be solved using quadratic programming techniques [Wu and Zhou, 2005].

$$w \cdot x_i - b \geq +1 \text{ for } y_i = +1 \tag{2.15}$$

$$w \cdot x_i - b \leq -1 \text{ for } y_i = -1 \tag{2.16}$$

These techniques can be extended for solving non-separable problems, as the ones

shown in Figure 2.8, by either introducing soft margin hyperplanes [Cortes and Vapnik, 1995] (i.e., hyperplanes allowing some training errors), or by using a kernel trick [Aizerman et al., 1964] which maps the inputs to a higher dimensional space, and the corresponding instances in this (new) space become linearly separable.

An interesting property of SVMs is that the decision surface is determined only by the examples which have exactly the distance $\frac{1}{|w|}$ from the decision plane. Those instances are called the support vectors, which are the only useful instances in the training data; if all other instances were removed, the same decision function will be learned.

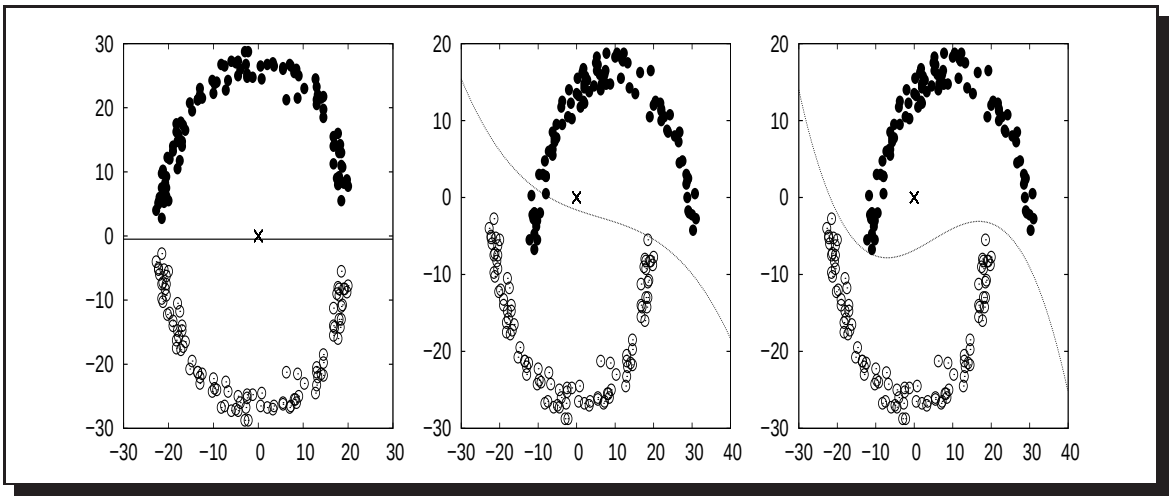

Figure 2.8. Mapping functions with increasing complexity.

2.6 Theoretical and Practical Remarks

Next we discuss some important remarks in classification.

2.6.1 The Need for Bias

From all possible mapping functions in $\mathcal{H}$, selecting an accurate one is a challenge. Any factors other than the provided examples that determine the mapping function that will be produced by the classification algorithm are called the *bias* of the algorithm. That is, the bias of a classification algorithm is essentially the set of assumptions used to determine the mapping function to be produced. Thus, classification algorithms may present different performances on the test set, depending on the bias which is implicit in their design. The following is a list of common biases employed by different classification algorithms:

- Maximum Margin: when drawing a boundary separating two outputs, attempt to maximize the width of the boundary. This is the bias used by SVM algorithms [Boser et al., 1992]. The assumption is that distinct outputs tend to be separated by wide boundaries.

- Minimum Description Length: when building a mapping function, attempt to minimize the length of the description of the function. This is the bias used to prune decision trees [Mehta et al., 1996; Fayyad and Irani, 1990]. The assumption is that simpler functions are more likely to be accurate.

- Nearest Neighbors: given an input, guess that its output is the same output related to the majority in its immediate neighborhood in feature space. This is the bias used in kNN classification algorithms [Dasarathy, 1990]. The assumption is that inputs in a small neighborhood in feature space are related to the same output.

Each classification algorithm encodes specific assumptions about how the optimal algorithm would be and works best when this assumption is satisfied by the problem to which it is applied.

2.6.2 No Free Lunch

The "No Free Lunch" theorem [Wolpert, 1996; Schaffer, 1994] states that all classification algorithms have identical performance over the set of all possible classification problems. Specifically, if a classification algorithm performs particularly well in a problem, there must exist a dual problem in which the same algorithm performs particularly badly. Thus, classification algorithms cannot be universally good. Further, on average, no algorithm is better than randomly guessing outputs.

The major criticism of the "No Free Lunch" theorems is that they are proven under the most general conditions possible. For instance, they assume that all the configurations of a classification problem are equaly likely to occur, while, in practice, some configurations are not even expected to occur (i.e., if a configuration is likely to occur in practice, its dual configuration is very unlikely to occur). Thus, it is important to develop classification algorithms that perform particularly well on problems that are likely to occur in practice. This highlights the need for bias while developing classification algorithms.

Chapter 3

Associative Classification

The hypothesis space, $\mathcal{H}$, may contain a huge (possibly infinite) number of functions. Randomly producing functions, in the hope of finding one that approximates well the target function $\mathrm{P}(y|x)$, is not likely to be an efficient strategy. Fortunately, there are countless more efficient strategies for producing approximations of $\mathrm{P}(y|x)$. One of these strategies is to directly exploit relationships, dependencies and associations between inputs and outputs (i.e., classes) [Liu et al., 1998]. Such associations are usually hidden in the examples in $\mathcal{S}$, and, when uncovered, they may reveal important aspects concerning the underlying phenomenon that generated these examples (i.e., $\mathrm{P}(y|x)$). These aspects can be exploited for sake of producing only functions that provide potentially good approximations of $\mathrm{P}(y|x)$ [Li et al., 2001; Cheng et al., 2007, 2008; Fan et al., 2008]. This strategy has led to a new family of classification algorithms which are often referred to as associative classification algorithms. The mapping functions produced by these algorithms are composed of rules $\mathcal{X} \rightarrow c_j$, indicating an association between $\mathcal{X}$, which is a set of attribute-values, and a class $c_j \in y$. In the following sections we will discuss some preliminaries and ways to extract rules from $\mathcal{S}$. Then, novel associative classification algorithms are presented and evaluated.

3.1 Preliminaries

We first present some discretization methods. Then, we discuss rules associating discretized attribute-values (which are subsets of an input) and classes (outputs).

3.1.1 Discretization

Attributes can assume real or nominal values. In the case of real-valued attributes, the number of possible attribute-values is virtually infinite. This causes undesirable

consequences for associative classification algorithms, due to high dimensionality (i.e., huge hypothesis space). An immediate question is whether it is possible to reduce the dimensionality by grouping together attribute-values. Discretization refers to the process of splitting the space of attribute-values into intervals, where each interval contains values that carry (almost) the same information. The crucial point during discretization is the definition of the boundaries of each interval. After discretization, real-valued and nominal attributes are treated indistinctly, and discretized attribute-values are referred to as features.

Uniform Range and Uniform Frequency Discretization

A simple discretization method is to split the space of values of each attribute into equal, predefined interval ranges. This method is known as Uniform Range Discretization. Figure 3.1(Left) shows the intervals defined by this method. Another simple strategy is to split the space of values of each attribute into variable-size intervals, so that each interval contains approximately the same number of examples. This method is known as Uniform Frequency Discretization. Figure 3.1(Middle) shows the intervals defined by this method.

Since these methods do not use information about the input-output relationship in order to set interval boundaries, it is likely that important information will be lost as a result of combining values that are strongly associated with different classes into the same interval. This may be harmful to classification effectiveness.

Discretization based on Minimum Description Length

A more sophisticated discretization method was proposed by Fayyad and Irani [1993]. It recursively splits the space of values of each attribute. The boundaries are those that provide the maximum information gain, that is, the intervals are found in a way that minimizes the entropy of the classes (e.g., intervals tend to include examples of the prevailing class). Splitting continues until the gain is below the minimal description length of the interval. This may result in an arbitrary number of intervals, including a single interval in which case the attribute is discarded as useless. Figure 3.1(Right) shows the intervals defined by this method.

3.1.2 Association Rules and Decision Rules

Association rules are implications of the form $a \rightarrow b$, where a and b are conjunctions of features/classes (i.e., any non-empty combination of features or classes). A much

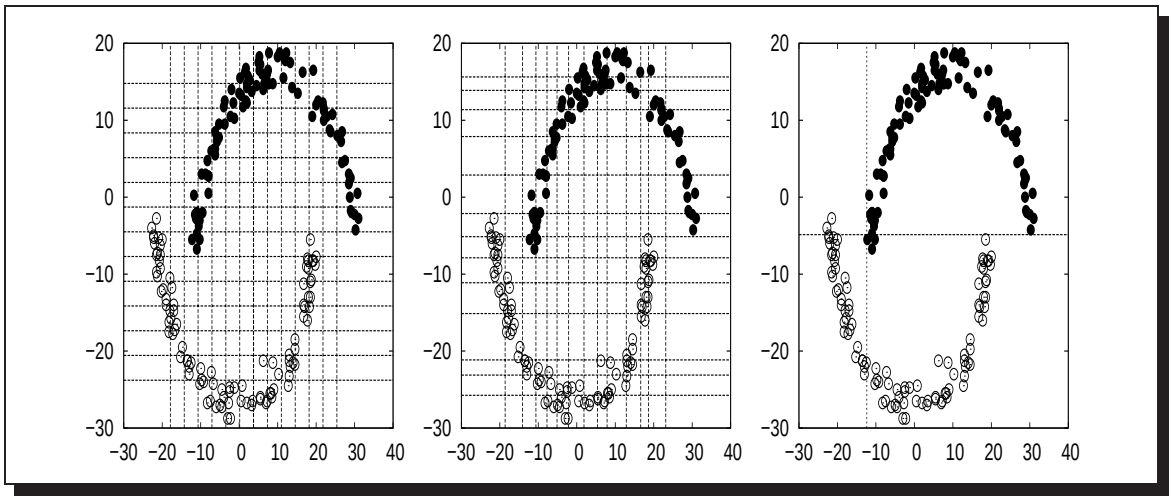

Figure 3.1. Discretized input space.

smaller subset of all association rules are particularly interesting for sake of approximating $\mathrm{P}(y|x)$. This subset is composed of rules, which hereafter will be called decision rules. Decision rules are implications of the form $\mathcal{X} \rightarrow c_j$, where $\mathcal{X}$ is any combination of features, and $c_j \in y$ is a class. Thus, these implications can be considered local mappings from inputs to outputs.

Decision rules are extracted from $\mathcal{S}$. Thus, a decision rule $\mathcal{X} \rightarrow c_j$ only exists if the set of features $\mathcal{X}$ is a subset of at least one input $x_i \in \mathcal{S}$. Next we discuss some important properties of decision rules.

Confidence. A decision rule $\mathcal{X} \rightarrow c_j$ is only interesting for sake of approximating $\mathrm{P}(y|x)$ if $\mathcal{X}$ and c_j are somehow associated. The confidence of a decision rule $\mathcal{X} \rightarrow c_j$, which is denoted as $\theta(\mathcal{X} \rightarrow c_j)$, is an indication of how strongly $\mathcal{X}$ and c_j are associated. It is given by the fraction of inputs containing $\mathcal{X}$ as subset, for which the corresponding output is c_j. Formally:

$$\theta(\mathcal{X} \rightarrow c_j) = \frac{|(x_i, y_i)| \in \mathcal{S} \;\text{ such that }\; \mathcal{X} \subseteq x_i \;\text{ and }\; c_j = y_i}{|(x_i, y_i)| \in \mathcal{S} \;\text{ such that }\; \mathcal{X} \subseteq x_i} \tag{3.1}$$

Support. A decision rule is only interesting for sake of approximating $\mathrm{P}(y|x)$ if the information it provides is reliable. The support of a decision rule $\mathcal{X} \rightarrow c_j$, which is denoted as $\sigma(\mathcal{X} \rightarrow c_j)$, is an important indication of the reliability of the association between $\mathcal{X}$ and c_j. It is given by the fraction of inputs in $\mathcal{S}$ containing $\mathcal{X}$ as a subset, and having c_j as the corresponding output. Formally:

$$\sigma(\mathcal{X} \rightarrow c_j) = \frac{|(x_i, y_i)| \in \mathcal{S} \;\text{ such that }\; \mathcal{X} \subseteq x_i \;\text{ and }\; c_j = y_i}{n} \tag{3.2}$$

Complexity. There is no universal way of measuring complexity. Often, however, the length of an explanation can be an indication of its complexity [Kolmogorov, 1965; Solomonoff, 1964; Chaitin, 1969; Rissanen, 1978]. A decision rule $\mathcal{X} \rightarrow c_j$ can be viewed as an explanation that leads to the decision c_j. In this case, the explanation is essentially the features in $\mathcal{X}$. Feature sets composed of few features provide simple explanations about the decision, while more complex explanations require more features. The size of the decision rule is the number of features in $\mathcal{X}$ (i.e., $|\mathcal{X}|$) and is regarded as the complexity of the corresponding explanation.

Usefulness. A decision rule $\mathcal{X} \rightarrow c_j$ matches an input $x_i \in \mathcal{T}$ if and only if $\mathcal{X} \subseteq x_i$. Since the prediction provided by a decision rule is based on all the features that are in $\mathcal{X}$, only decision rules matching input x_i are used to estimate the corresponding output y_i. A decision rule is useful for sake of approximating $\mathrm{P}(y|x)$ if it matches at least one input in $\mathcal{T}$, otherwise the decision rule is said to be useless. For practical problems involving many features, useful decision rules correspond to a very small subset of all possible decision rules.

3.2 Method and Algorithms

In this section a simple method for associative classification will be presented. Similar methods were already proposed in [Li et al., 2001; Liu et al., 1998]. First, it will be described how decision rules are extracted from $\mathcal{S}$, and then how these decision rules are used in order to approximate $\mathrm{P}(y|x)$. The method to be presented will hereafter be referred to as EAC (standing for Eager Associative Classification). Algorithms based on this method will also be presented in this section.

3.2.1 Level-Wise Rule Extraction

Simpler decision rules are extracted before more complex ones. Specifically, decision rules of size k are extracted from $\mathcal{S}$ before decision rules of size $k+1$. The set of rules of size k is denoted as $\mathcal{F}_k$. The support of a decision rule $\mathcal{X} \rightarrow c_j$ is calculated by directly accessing $\mathcal{S}$ and counting the number of inputs having $\mathcal{X}$ as subset, for which c_j is the corresponding output.

Support Counting

Efficient support counting is mandatory for the scalability of the algorithm. The main objective is to reduce the number of accesses to $\mathcal{S}$. The algorithms to be presented in

this thesis employ an efficient strategy for support counting, which will be described next.

Vertical Bitmap Representation. A vertical bitmap is created for each feature and class (or output) in $\mathcal{S}$. That is, $\mathcal{S}$ is translated into a bitmap matrix, in which the columns are indexed by features (or classes), and lines are indexed by examples. If feature (or class) a_i appears in example s_j, then the bit indexed by example s_j and feature (or class) a_i is set to one; otherwise, the bit is set to zero.

Bit-Wise Operations. Given the bitmap for feature a_i, and the bitmap for feature a_j, the bitmap for feature-set $\{a_i, a_j\}$ is simply the bitwise AND of these two bitmaps. Classes are processed in the same way. Each bitmap is divided into words, and each word is 16-bit length (i.e., each word comprises 16 examples).

Look-Up Table. All possible results of a bit-wise operation are stored in a look-up table with 2^{16} entries. After a word is processed, the corresponding result for this word is retrieved from the look-up table. The results associated with each word are finally summed. The final sum is the support count.

Pruning

The number of all possible decision rules hidden in $\mathcal{S}$ may be huge. Very often, practical limitations prevent the extraction of all such decision rules. Thus, pruning strategies must be employed in order to reduce the number of decision rules that are processed. EAC employs the typical pruning strategy which is to use a single cut-off value based on a minimum support threshold, σ_{min}. This cut-off value separates frequent from non-frequent decision rules. Specifically, only decision rules occurring at least $\sigma_{min} \times |\mathcal{S}|$ times in $\mathcal{S}$, are considered frequent. Since decision rules are extracted in a level-wise, bottom-up way, only feature-sets associated with frequent rules of size k are used to produce rules of size $k+1$ [Agrawal et al., 1993] (i.e., only feature-sets included in rules in $\mathcal{F}_k$ are used to produce rules in $\mathcal{F}_{k+1}$).

3.2.2 Prediction

Mapping functions perform predictions using the decision rules extracted from $\mathcal{S}$, as it will be discussed next.

Grouping

After extracting decision rules from $\mathcal{S}$, EAC groups them in order to facilitate fast access to specific ones. In the discussion that follows it will be denoted as $\mathcal{R}$ an arbitrary set of decision rules extracted from $\mathcal{S}$ (i.e., frequent rules in $\mathcal{S}$). Similarly, it will be denoted as $\mathcal{R}^{x_i}$ the subset of $\mathcal{R}$ which contains only the decision rules matching an arbitrary input x_i. Finally, it will be denoted as $\mathcal{R}^{x_i}_{c_j}$ the subset of $\mathcal{R}^{x_i}$ which contains only decision rules predicting class c_j (i.e., decision rules of the form $\mathcal{X} \rightarrow c_j$ for which $\mathcal{X} \subseteq x_i$).

Learning Mapping Functions

Given an arbitrary input x_i, $f_{\mathcal{S}}(x_i)$ gives the predicted output for x_i. The value of $f_{\mathcal{S}}(x_i)$ is calculated based on the decision rules in $\mathcal{R}^{x_i}$. Different algorithms based on the EAC method can be distinguished depending on the prediction strategy used.

EAC-SR – This algorithm (standing for EAC using a single rule) produces a mapping function, $f_{\mathcal{S}}$, using frequent decision rules in $\mathcal{S}$. Then, given an input x_i, $f_{\mathcal{S}}$ simply returns the output (or the class) predicted by the strongest decision rule (i.e., the one with highest confidence value) matching x_i, that is:

$$f_{\mathcal{S}}(x_i) = c_j \text{ such that } \theta(\mathcal{X} \rightarrow c_j) \text{ is } \operatorname{argmax}(\theta(r)) \; \forall \; r \in \mathcal{R}^{x_i} \tag{3.3}$$

EAC-SR is a PAC-efficient classification algorithm, as will be demonstrated next.

Theorem 3.1. *EAC-SR is PAC-efficient.*

Proof. Let q be the number of features in $\mathcal{S}$. In this case, the hypothesis space $\mathcal{H}$ for EAC-SR is the set of all possible decision rules in $\mathcal{S}$, which is clearly $|\mathcal{H}| = p \times 2^q$. According to Equation 2.4

$$\begin{aligned} n &\geq \frac{1}{\epsilon}\left(\ln(p \times 2^q) + \ln(\frac{1}{\delta})\right) \\ &\geq \frac{1}{\epsilon}\left(\ln(p) + 0.69q + \ln(\frac{1}{\delta})\right) \end{aligned}$$

Thus, the sample complexity increases only polynomially with q. It is also necessary to show that a function $f_{\mathcal{S}}$, for which $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] = 0$ is found in time polynomial in q.

Obviously, the number of functions in $\mathcal{S}$ is exponential in q (i.e., $O(2^q)$). However, since an input is composed of l attribute-values (or features), the maximum allowed size of decision rules can be held fixed to l (i.e., a decision rule can have at most l features).

Thus, the number of possible decision rules in $\mathcal{S}$ is $p \times (q + \binom{q}{2} + \ldots + \binom{q}{l}) = O(q^l)$, and, thus, $f_{\mathcal{S}}$ can be found in time polynomial in q, according to Algorithm 1. This completes the proof. □

Algorithm 1 Finding $f_{\mathcal{S}}$, according to EAC-SR.

Require: The training data $\mathcal{S}$, σ_{min}, and l
Ensure: $f_{\mathcal{S}}$, if it exists. Failure, otherwise.

1: $\mathcal{R} \Leftarrow$ rules $\mathcal{X} \rightarrow c_j$ extracted from $\mathcal{S}$, such that $|\mathcal{X}| \leq l$ and $\sigma(\mathcal{X} \rightarrow c_j) \geq \sigma_{min}$
2: **if** $\forall x_i \in \mathcal{S}\ \exists r \in \mathcal{R}^{x_i}$ such that $\theta(r) = 1.00$ **then**
3: **r**eturn $f_{\mathcal{S}}$ such that $f_{\mathcal{S}}(x_i) = c_j$, where $\mathcal{X} \rightarrow c_j \in \mathcal{R}^{x_i}$ and $\theta(\mathcal{X} \rightarrow c_j) = 1.00$
4: **else**
5: **r**eturn failure.
6: **end if**

An immediate question is to find under which circumstances $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] = 0$.

Theorem 3.2. *If* $\sigma_{min} \leq \frac{1}{|\mathcal{S}|}$, *then* $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] = 0$.

Proof. If $\sigma_{min} \leq \frac{1}{|\mathcal{S}|}$ then any feature set occurring at least once in $\mathcal{S}$ is frequent. Since $\mathcal{S}$ is composed of distinct inputs (i.e., instances with different features sets), there must be feature sets occurring only once in $\mathcal{S}$. A feature set, $\mathcal{X}$, occurring once in $\mathcal{S}$ (an extreme case is $\mathcal{X} = x_i$) must produce a decision rule $\mathcal{X} \rightarrow c_j$ such that $\theta(\mathcal{X} \rightarrow c_j) = 1.00$. Further, for any feature set $\mathcal{Y}$ such that $\mathcal{Y} \subseteq \mathcal{X}$, if there is a decision rule $\mathcal{Y} \rightarrow \overline{c_j}$ then $\theta(\mathcal{Y} \rightarrow \overline{c_j}) < 1.00$. Therefore, rule $\mathcal{Y} \rightarrow \overline{c_j}$ is never the strongest one (since rule $\mathcal{X} \rightarrow c_j$ is always stronger than it), an thus $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] = 0$. □

Theorem 3.2 gives a lower bound for σ_{min} which ensures that $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] = 0$. Unfortunately, in real-world scenarios, setting σ_{min} to $\frac{1}{|\mathcal{S}|}$, is not practical, since the number of decision rules that will be generated is overwhelming. In practice, as will be empirically shown in Section 3.4.2, $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] \rightarrow 0$ when $\sigma_{min} \rightarrow \frac{1}{|\mathcal{S}|}$.

Next we present an example which illustrates the basic steps of EAC-SR.

Example. Consider Table 3.1. There are 10 pairs in $\mathcal{S}$. All inputs were discretized. Suppose σ_{min} is set to 0.30. In this case:

- $\mathcal{F}_1$ contains:
  1. $\{a_1 =$[0.00-0.22]$\rightarrow output = 1\}$ $(\theta = 0.75)$
  2. $\{a_2 =$[0.33-0.71]$\rightarrow output = 1\}$ $(\theta = 0.80)$
  3. $\{a_3 =$[0.00-0.35]$\rightarrow output = 1\}$ $(\theta = 1.00)$
  4. $\{a_3 =$[0.35-1.00]$\rightarrow output = 0\}$ $(\theta = 0.60)$

		Input (x_i)			Output (y_i)
		a_1	a_2	a_3	
$\mathcal{S}$	(x_1, y_1)	[0.00-0.22]	[0.33-0.71]	[0.00-0.35]	1
	(x_2, y_2)	[0.00-0.22]	[0.33-0.71]	[0.35-1.00]	0
	(x_3, y_3)	[0.46-1.00]	[0.71-1.00]	[0.35-1.00]	1
	(x_4, y_4)	[0.22-0.46]	[0.00-0.33]	[0.35-1.00]	0
	(x_5, y_5)	[0.00-0.22]	[0.33-0.71]	[0.00-0.35]	1
	(x_6, y_6)	[0.22-0.46]	[0.33-0.71]	[0.00-0.35]	1
	(x_7, y_7)	[0.00-0.22]	[0.33-0.71]	[0.00-0.35]	1
	(x_8, y_8)	[0.22-0.46]	[0.71-1.00]	[0.00-0.35]	1
	(x_9, y_9)	[0.46-1.00]	[0.00-0.33]	[0.35-1.00]	1
	(x_{10}, y_{10})	[0.22-0.46]	[0.00-0.33]	[0.35-1.00]	0
$\mathcal{T}$	(x_{11}, y_{11})	[0.22-0.46]	[0.33-0.71]	[0.35-1.00]	?(1)

Table 3.1. Training data and test set given as example.

- $\mathcal{F}_2$ contains:

  1. $\{a_1 =$[0.00-0.22]$\wedge a_2 =$[0.33-0.71]$\rightarrow output = 1\}$ $(\theta = 0.75)$
  2. $\{a_1 =$[0.00-0.22]$\wedge a_3 =$[0.00-0.35]$\rightarrow output = 1\}$ $(\theta = 1.00)$
  3. $\{a_2 =$[0.33-0.71]$\wedge a_3 =$[0.00-0.35]$\rightarrow output = 1\}$ $(\theta = 1.00)$

- $\mathcal{F}_3$ contains:

  1. $\{a_1 =$[0.00-0.22]$\wedge a_2 =$[0.33-0.71]$\wedge a_3 =$[0.00-0.35]$\rightarrow output = 1\}$ $(\theta = 0.75)$

- $\mathcal{F}_4 = \emptyset$, and no more frequent decision rules can be extracted from $\mathcal{S}$.

Clearly, $\mathcal{R} = \{\mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3\}$. There is one input in $\mathcal{T}$, x_{11}, for which the corresponding output, y_{11}, is unknown. The selected function $f_{\mathcal{S}}$ will use the rule set $\mathcal{R}^{x_{11}}$ in order to predict such output. $\mathcal{R}^{x_{11}}$ contains only 2 rules:

1. $\{a_2 =$[0.33-0.71]$\rightarrow output = 1\}$ $(\theta = 0.80)$
2. $\{a_3 =$[0.35-1.00]$\rightarrow output = 0\}$ $(\theta = 0.60)$

According to Equation 3.3, EAC-SR simply picks the strongest rule in $\mathcal{R}^{x_{11}}$, and thus it predicts output 1 for input x_{11}.

In the remaining of this thesis we propose several improvements to EAC-SR. Such improvements lead to other classification algorithms, which are more efficient and practical. Thus, EAC-SR is a starting point for the classification algorithms proposed in this thesis.

EAC-MR – A single decision rule is simply a local mapping of parts of some inputs to an output, and thus it only provides a fragmented, incomplete information about $P(y|x)$. This makes EAC-SR prone to error, since it picks a single, very strong decision rule to perform predictions. Such a simple pick may provide biased predictions, and is likely to suffer from overfitting (very strong rules tend to be too specific). A safer strategy is to produce a global mapping by combining the predictions of multiple decision rules, so that a collective prediction can be performed. Intuitively, that would help avoiding bias and overfitting [Li et al., 2001].

Given input x_i, EAC-MR (standing for EAC using multiple rules) produces a mapping function $f_{\mathcal{S}}$, which returns the output (or the class) which receives the highest score in a weighted voting process. Specifically, $\mathcal{R}^{x_i}$ is interpreted as a poll, in which each decision rule $\mathcal{X} \rightarrow c_j \in \mathcal{R}^{x_i}$ is a vote given by $\mathcal{X}$ for output c_j. The weight of a vote $\mathcal{X} \rightarrow c_j$ depends on $\theta(\mathcal{X} \rightarrow c_j)$. The score associated with output c_j for input x_i, denoted as $s(x_i, c_j)$, is:

$$s(x_i, c_j) = \frac{\sum_{r \in \mathcal{R}_{c_j}^{x_i}} \theta(r)}{\mid \mathcal{R}_{c_j}^{x_i} \mid} \tag{3.4}$$

The likelihood of class c_j being the output of input x_i, denoted as $\hat{p}(c_j|x_i)$, is:

$$\hat{p}(c_j|x_i) = \frac{s(x_i, c_j)}{\sum_{k=1}^{p} s(x_i, c_k)} \tag{3.5}$$

where p is the number of possible outputs (i.e., the number of distinct classes in $\mathcal{S}$). Finally:

$$f_{\mathcal{S}}(x_i) = c_j \text{ such that } \hat{p}(c_j|x_i) \text{ is argmax } (\hat{p}(c_k|x_i)), \text{ where } 1 \leq k \leq p \tag{3.6}$$

The basic steps of EAC-MR are shown in Algorithm 2.

Algorithm 2 Finding $f_{\mathcal{S}}$, according to EAC-MR.

Require: The training data $\mathcal{S}$, and σ_{min}, and l
Ensure: $f_{\mathcal{S}}$.

1: $\mathcal{R} \Leftarrow$ rules $\mathcal{X} \rightarrow c_j$ extracted from $\mathcal{S}$, such that $|\mathcal{X}| \leq l$ and $\sigma(\mathcal{X} \rightarrow c_j) \geq \sigma_{min}$
2: **r**eturn $f_{\mathcal{S}}$ such that $f_{\mathcal{S}}(x_i) = c_j$, where $\hat{p}(c_j|x_i)$ is argmax$(\hat{p}(c_k|x_i)) \; \forall \; 1 \leq k \leq p$

3.2.3 Function Approximation

While the main objective of EAC-SR is to learn a function $f_{\mathcal{S}}$ for which $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] = 0$ (i.e., consistency with training data), EAC-MR aims at learning a function $f_{\mathcal{S}}$ for which $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] \approx \mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$ (i.e., generalization). While Theorems 3.1 and 3.2 state that EAC-SR produces functions that are consistent with the training data (i.e., $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] = 0$), EAC-MR must rely on specific function approximation strategies in order to learn functions that are more likely to generalize.

EAC-MR may employ two function approximation techniques, which were described in Section 2.3. This leads to two new algorithms, EAC-MR-ERM, which employs the well known Empirical Risk Minimization technique to approximate the target function, and EAC-MR-SRM, which employs the Structural Risk Minimization technique to approximate the target function. Both algorithms will be described next.

EAC-MR-ERM – A simple strategy to empirical risk minimization is to include only highly complex, very long decision rules in $\mathcal{R}$. In this case, a function $f_{\mathcal{S}}$ which makes predictions according to rules in $\mathcal{R}$ will fit $\mathcal{S}$ almost perfectly. The obvious problem is overfitting, because too complex decision rules may be as long as the training data itself, and thus $f_{\mathcal{S}}$ is unlikely to generalize. A likely to generalize function, on the other hand, would use decision rules which are considerably shorter than the training data. The problem, in this case, is underfitting, since too simple decision rules may not capture the underlying properties of $\mathcal{S}$. Figure 3.2 illustrates this idea. High-order polynomials are able to fit $\mathcal{S}$, while low-order polynomials are unable to properly fit $\mathcal{S}$. The same trend can be observed in mapping functions that use either very complex or very simple decision rules.

Selecting a function $f_{\mathcal{S}}$ with the appropriate complexity is challenging, but mandatory for effective classification. An approach to control the complexity of $f_{\mathcal{S}}$ is to establish a relationship between how well $f_{\mathcal{S}}$ fits $\mathcal{S}$ (i.e., how close $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}]$ is to 0), and how much $f_{\mathcal{S}}$ is dependent on $\mathcal{S}$. If variations in $\mathcal{S}$ (i.e., removing a pair) do not considerably change $f_{\mathcal{S}}$, then $f_{\mathcal{S}}$ is likely to fit pairs that are not in $\mathcal{S}$ as well as it fits pairs that are in $\mathcal{S}$ (i.e., $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] \approx \mathcal{I}_{\mathcal{T}}[f_{\mathcal{S}}]$). If, additionally, $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}]$ is close to 0, then $f_{\mathcal{S}}$ effectively approximates $\mathrm{P}(y|x)$. The stability [Bousquet and Elisseeff, 2002; Kutin and Niyogi, 2002; Mukherjee et al., 2006] of a function, which is denoted as β, measures at which extent small variations on $\mathcal{S}$ change $f_{\mathcal{S}}$.

One way to assess the stability of a function $f_{\mathcal{S}}$ is to compare $f_{\mathcal{S}}(x_i)$ with $f_{\mathcal{S}-s_i}(x_i)$ at each pair $s_i = (x_i, y_i) \in \mathcal{S}$ (where $f_{\mathcal{S}-s_i}$ is the function selected from $\{\mathcal{S} - s_i\}$, that is, the training data after the specific pair s_i is removed). According to Equation 2.8, the stability of $f_{\mathcal{S}}$ is given by the highest value of $|f_{\mathcal{S}}(x_i) - f_{\mathcal{S}-s_i}(x_i)|$ over all pairs in

$\mathcal{S}$. Stability can be used to select a function with appropriate complexity.

Selecting $f_{\mathcal{S}}$ according to Stability. The hypothesis space is traversed by evaluating candidate functions in increasing order of complexity. That is, simpler functions are produced before more complex ones. To achieve this, we introduce a nested structure of subsets of rules, $h_1, h_2, \ldots, h_l$, where h_i contains frequent decision rules $\mathcal{X} \rightarrow c_j$ for which $|\mathcal{X}| \leq l$. Clearly, $h_1 \subseteq h_2 \subseteq \ldots \subseteq h_l$. The simplest candidate function is the one which uses decision rules in h_1 (i.e., this function uses only decision rules of size 1), while the most complex one uses decision rules in h_l. From these candidate functions, we select the one which minimizes Equation 2.9. This selection process involves a trade-off: as the complexity of the rules increases the empirical error decreases, but the stability of the corresponding function also decreases. Figure 3.3 illustrates this trade-off. It shows functions obtained before and after removing the black point. Figure in the left shows the case in which $f_{\mathcal{S}}$ is stable (i.e., low variation between $f_{\mathcal{S}}$ and $f_{\mathcal{S}-s_i}$, where s_i is the black point) but it does not fit $\mathcal{S}$ well. Figure in the middle shows the desirable case, in which $f_{\mathcal{S}}$ fits $\mathcal{S}$ relatively well and is stable. Figure in the right shows the case in which $f_{\mathcal{S}}$ fits $\mathcal{S}$ very well, but it is not stable anymore. The selected function is the one that best trades empirical error and stability. Algorithm 3 shows basic steps of EAC-MR-ERM.

Algorithm 3 Finding $f_{\mathcal{S}}$, according to EAC-MR-ERM.

Require: The training data $\mathcal{S}$, σ_{min}, l,and δ
Ensure: $f_{\mathcal{S}}$.

1: *tighest bound* $\Leftarrow \infty$
2: **for** $i = 1$ to l **do**
3: $\quad \mathcal{R} \Leftarrow h_i \Leftarrow$rules $\mathcal{X} \rightarrow c_j$, such that $|\mathcal{X}| \leq i$ and $\sigma(\mathcal{X} \rightarrow c_j) \geq \sigma_{min}$
4: $\quad$ **for** each pair $s_i = (x_i, y_i) \in \mathcal{S}$ **do**
5: $\quad\quad \beta_{s_i} = |f_{\mathcal{S}}(x_i) - f_{\mathcal{S}-s_i}(x_i)|$
6: $\quad$ **end for**
7: $\quad \beta = \sup(\beta_{s_i})$
8: $\quad bound \Leftarrow \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] + \left(\beta + (4n\beta + 1) \times \sqrt{\frac{\ln \frac{1}{\delta}}{2n}}\right)$
9: $\quad$ **if** *tighest bound* $\leq$ *bound* **then** break
10: $\quad$ *tighest bound* $\Leftarrow$ *bound*
11: **end for**
12: **return** $f_{\mathcal{S}}$ such that $f_{\mathcal{S}}(x_i) = c_j$, where $\hat{p}(c_j|x_i)$ is argmax$(\hat{p}(c_k|x_i))$ $\forall$ $1 \leq k \leq p$

EAC-MR-SRM – Another way of controlling the complexity of $f_{\mathcal{S}}$ is to establish a relationship between how well $f_{\mathcal{S}}$ fits $\mathcal{S}$, and how complex is $f_{\mathcal{S}}$. In Figure 3.2, where the x-axis represents the input space, and the y-axis represents the output space, the

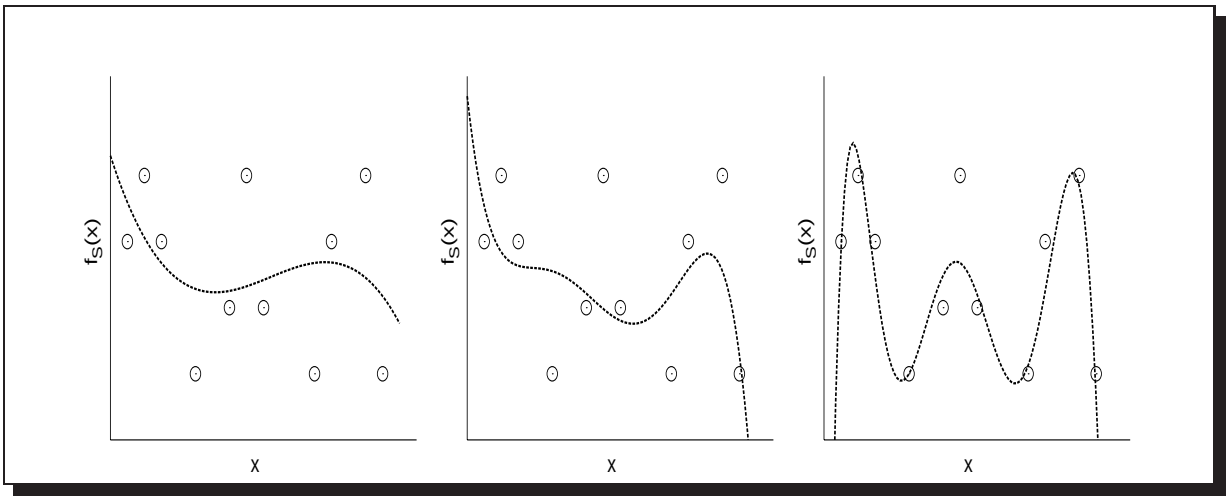

Figure 3.2. Polynomials of increasing degrees.

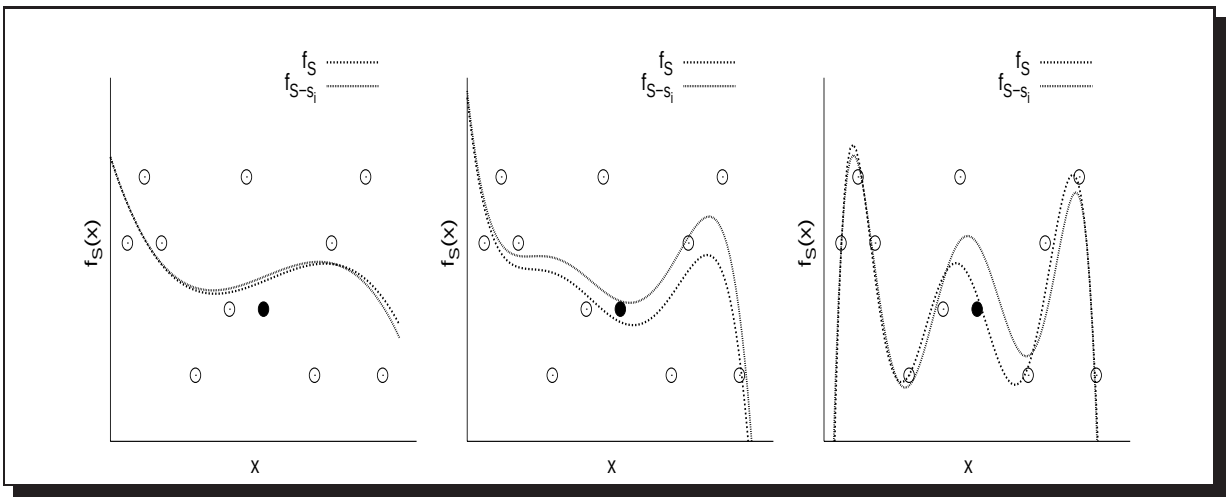

Figure 3.3. Trading-off complexity and stability.

complexity of a function is given by the number of free parameters (i.e., the polynomial degree). A more general measure of the complexity of a mapping function $f_{\mathcal{S}}$ is its VC-dimension, denoted as $d_{f_{\mathcal{S}}}$. Ideally, $f_{\mathcal{S}}$ has small VC-dimension and $\mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}]$ is close to 0.

Selecting $f_{\mathcal{S}}$ according to VC-dimension. The hypothesis space is traversed by evaluating candidate functions in increasing order of complexity. That is, simpler functions are produced before more complex ones. Again, it is employed a nested structure of subsets of rules, $h_1, h_2, \ldots, h_l$, where h_i contains frequent decision rules $\mathcal{X} \rightarrow c_j$ for which $|\mathcal{X}| \leq l$. From these candidate functions, the selected is the one which minimizes Equation 2.10. This selection process involves a trade-off: as the complexity of the rules increases the empirical error decreases, but the VC-dimension

of the corresponding function increases. The selected function is the one that best trades empirical error and VC-dimension. Algorithm 4 shows the detailed steps of EAC-MR-SRM.

Algorithm 4 Finding $f_{\mathcal{S}}$, according to EAC-MR-SRM.

Require: The training data $\mathcal{S}$, σ_{min}, and δ
Ensure: $f_{\mathcal{S}}$.

1: $tighest\ bound \Leftarrow \infty$
2: **for** $i = 1$ to l **do**
3: $\quad \mathcal{R} \Leftarrow h_i \Leftarrow$rules $\mathcal{X} \rightarrow c_j$, such that $|\mathcal{X}| \leq i$ and $\sigma(\mathcal{X} \rightarrow c_j) \geq \sigma_{min}$
4: $\quad d_{f_{\mathcal{S}}} = n \times (1 - \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}])$
5: $\quad bound \Leftarrow \mathcal{I}_{\mathcal{S}}[f_{\mathcal{S}}] + \sqrt{\frac{d_{f_{\mathcal{S}}}\left(\ln \frac{2n}{d_{f_{\mathcal{S}}}}+1\right)-\ln \frac{\delta}{4}}{n}}$
6: $\quad$ **if** $tighest\ bound \leq bound$ **then** break
7: $\quad tighest\ bound \Leftarrow bound$
8: **end for**
9: **return** $f_{\mathcal{S}}$ such that $f_{\mathcal{S}}(x_i) = c_j$, where $\hat{p}(c_j|x_i)$ is argmax$(\hat{p}(c_k|x_i))\ \forall\ 1 \leq k \leq p$

3.3 Empirical Results

In this section we will present the experimental results for the evaluation of the proposed associative classification algorithms, which include: EAC-SR, EAC-MR, EAC-MR-ERM, and EAC-MR-SRM.

Setup Continuous attributes were discretized using the MDL-based entropy minimization method [Fayyad and Irani, 1993], which was described in Section 3.1.1. In all experiments we used 10-fold cross-validation and the final results of each experiment represent the average of the ten runs. All results to be presented were found statistically significant based on a t-test at 95% confidence level.

Computational Environment The experiments were performed on a Linux-based PC with a Intel Pentium III 1.0 GHz processor and 1 GB RAM.

3.3.1 The UCI Benchmark

The UCI benchmark[1] provides a method for comparing the classification performance of various classification algorithms. We used a set of 26 datasets, obtained from various different applications. In this section we will evaluate the proposed algorithms using these datasets.

[1]Available at `http://archive.ics.uci.edu/ml/`

Baselines The evaluation is based on a comparison involving a decision tree algorithm (C4.5), a Naive Bayes algorithm (NB), and an SVM algorithm. For C4.5 and NB algorithms, we used the corresponding implementations available in MLC++ (Machine Learning Library in C++) proposed in [Kohavi et al., 1996]. For SVM, we used the implementation available at `http://svmlight.joachims.org/` (version 3.0). The well-known CBA associative classification algorithm Liu et al. [1998] (to be discussed in Section 3.4) was also used as baseline. Its implementation is available at `http://www.comp.nus.edu.sg/~dm2/` (version 2.1).

Parameters The implementation for Naive Bayes is non-parametric. For C4.5, we used the C4.5-auto-parm tool, available in MLC++, for finding optimum parameters for each dataset. For SVM, we used the grid parameter search tool in LibSVM [Chang and Lin, 2001] for finding the parameters for each dataset. For CBA, EAC-SR, EAC-MR, EAC-MR-ERM, and EAC-MR-SRM, we set σ_{min}=0.01 (we tried some values and selected the one which yields the best average performance using cross-validation on each training fold).

Evaluation Criteria Classification performance is expressed using the conventional true error rate in the test set.

Analysis Table 3.2 shows classification performance for different classification algorithms. Best results, including statistical ties, are shown in bold. EAC-SR outperforms C4.5 and NB. In fact, it was demonstrated in [Veloso et al., 2006b] that, if we set all algorithms under the information gain principle, an associative classification algorithm always outperforms the corresponding decision tree one. EAC-MR outperforms all baselines, showing the importance of employing multiple decision rules while approximating the target function. Furthermore, EAC-MR-ERM and EAC-MR-SRM were the best performers, suggesting that more sophisticated function approximation strategies are also capable of improving classification performance.

3.3.2 The ACM Digital Library

Organizing documents in order to facilitate fast access to information is a long-existing necessity. In the ancient times, the Library of Alexandria contained more than 120,000 scrolls, and, at that time, finding the desired information could take days. Callimachus of Cyrene is considered the first bibliographer, and is the one who proposed to organize the scrolls in the library by their subjects. He invented the first library catalog, a catalog of scrolls, which revolutionized the way people store information.

	EAC				Baselines			
Dataset	SR	MR	MR-ERM	MR-SRM	C4.5	NB	SVM	CBA
anneal	**0.025**	**0.025**	**0.025**	**0.025**	0.075	**0.027**	0.051	**0.021**
australian	**0.146**	**0.138**	**0.138**	**0.138**	0.148	**0.140**	**0.143**	**0.146**
auto	0.176	**0.156**	**0.156**	**0.156**	0.176	0.321	0.249	0.199
breast	0.106	0.074	0.074	0.074	0.056	**0.024**	**0.028**	0.037
cleve	0.142	**0.132**	**0.132**	**0.129**	0.215	0.171	0.166	0.171
crx	**0.127**	**0.127**	**0.123**	**0.127**	0.150	0.146	0.144	0.146
diabetes	0.261	0.254	0.244	**0.230**	0.261	0.244	**0.230**	0.255
german	0.265	0.256	**0.247**	**0.247**	0.284	**0.246**	0.288	0.265
glass	**0.255**	**0.255**	**0.253**	**0.251**	0.304	0.294	0.291	0.261
heart	0.168	**0.144**	**0.144**	**0.144**	0.218	0.181	**0.142**	0.181
hepatitis	0.183	0.183	0.167	**0.150**	0.182	**0.150**	0.167	0.189
horse	0.176	0.176	0.176	0.176	**0.147**	0.206	0.178	0.176
hypo	0.062	0.047	0.047	0.040	**0.007**	**0.015**	**0.013**	**0.001**
ionosphere	0.077	**0.063**	**0.063**	**0.063**	0.105	0.119	0.083	0.077
iris	0.081	**0.053**	0.067	**0.053**	**0.047**	0.060	**0.043**	**0.053**
labor	**0.033**	**0.033**	**0.033**	**0.033**	0.223	0.140	0.218	0.137
led7	0.325	0.281	0.267	**0.252**	0.305	0.267	**0.252**	0.281
lymph	0.180	**0.130**	**0.130**	**0.130**	0.238	0.244	0.197	0.221
pima	0.301	0.258	0.245	**0.230**	0.258	0.245	0.230	0.271
sick	0.061	0.061	0.061	0.061	**0.011**	0.039	0.032	0.028
sonar	0.220	**0.067**	**0.067**	**0.067**	0.284	0.230	0.160	0.225
tic-tac-toe	0.138	0.108	0.100	0.100	0.138	0.301	0.167	**0.004**
vehicle	0.315	0.315	0.310	0.310	0.285	0.401	**0.254**	0.310
waveform	0.225	0.225	0.219	0.219	0.228	0.193	**0.102**	0.203
wine	0.050	0.050	**0.00**	**0.00**	0.073	0.095	0.021	0.050
zoo	0.096	0.096	0.070	0.070	0.078	0.137	0.049	**0.032**
Avg	0.161	0.142	**0.137**	**0.134**	0.173	0.178	0.150	0.151

Table 3.2. Classification performance for different algorithms.

Currently, fast access to information is still a central issue in digital libraries. In order to build the catalog, it is necessary to effectively group documents by common topics or subjects − a task known as document categorization. The dominant approach to document categorization is based on the application of classification algorithms. In this case, for a document given as input, its subject (or category) must be given as output.

In this section we will evaluate the proposed algorithms using a collection of documents extracted from the ACM digital library[2]. The collection contains 6,682 documents, which were labeled using 8 first level categories of ACM, namely: Hardware, Computer Systems Organization, Software, Computing Methodologies, Mathematics

[2] `http://portal.acm.org/dl.cfm/`

of Computing, Information Systems, Theory of Computation, Computing Milieux. Citations and words in title/abstract of a document compose the corresponding set of features. The collection has a vocabulary of 9,840 unique words, and a total of 51,897 citations.

Baselines The evaluation is based on a comparison involving general-purpose algorithms, such as kNN and SVM. For kNN, we used the implementation available in MLC++ [Kohavi et al., 1996]. For SVM, we used the implementation available at `http://svmlight.joachims.org/` (version 3.0). Application-specific methods were also used in the evaluation. Amsler [Amsler, 1972] is a very used bibliographic-based method for document categorization. The Bayesian method [Calado et al., 2003] and Multi-Kernel SVMs [Joachims et al., 2001] are two state-of-the-art representatives of methods for document categorization.

Parameters The implementation for Amsler and Bayesian methods are non-parametric. For SVM and Multi-Kernel SVMs, we used the grid parameter search tool in LibSVM [Chang and Lin, 2001] for finding the optimum parameters. For kNN, we carefully tuned k=15. For EAC-MR, EAC-MR-ERM, and EAC-MR-SRM, we set σ_{min}=0.005.

Evaluation Criteria Categorization performance for the various methods being evaluated is expressed through F_1 measures. In this case, precision p is defined as the fraction of correctly classified documents in the set of documents classified as positive. Recall r is defined as the fraction of correctly classified documents out of all the documents having the target category. F_1 is a combination of precision and recall defined as the harmonic mean $\frac{2pr}{p+r}$. Macro- and micro-averaging [Yang et al., 2002] were applied to F_1 to get single performance values over all classification tasks. For F_1 macro-averaging ($MacF_1$), scores were first computed for individual categories and then averaged over all categories. For F_1 micro-averaging ($MicF_1$), the decisions for all categories were counted in a joint pool. The computational efficiency is evaluated through the total execution time, that is, the processing time spent in training and classifying all documents.

Analysis Table 3.3 shows categorization performance for various classification algorithms. Amsler was used as the baseline for comparison. The Multi-Kernel algorithm achieved the highest values of $MicF_1$ and $MacF_1$. Associative classification algorithms were not very effective in this application scenario, mainly due to the large number of features, which leads to a huge number of decision rules.

Algorithms	MicF_1	MacF_1	Gains (%) over baseline		Execution Time
			MicF_1	MacF_1	
Amsler (baseline)	0.832	0.783	–	–	1,251 secs
EAC-MR	0.766	0.692	-0.079	-0.115	2,350 secs
EAC-MR-ERM	0.789	0.736	-0.051	-0.060	2,921 secs
EAC-MR-SRM	0.812	0.767	-0.024	-0.020	2,419 secs
kNN	0.833	0.774	0.001	-0.011	**83** secs
SVM	0.845	**0.810**	0.016	0.035	1,932 secs
Bayesian	0.847	0.796	0.019	0.016	8,281 secs
Multi-Kernel	**0.859**	**0.812**	0.032	0.037	14,894 secs

Table 3.3. Categorization performance for different algorithms.

We analyzed the discrepancy between rule confidence values in $\mathcal{S}$ and $\mathcal{T}$. If the discrepancy is low, then the algorithm is likely to be effective. Figure 3.4 shows discrepancy values as a function of rule support. In this case, each point corresponds to a rule. Lighter colored regions represent rules with higher support values. Rules with low discrepancy in confidence appear in the diagonal. As it can be seen, rules with low discrepancy have usually higher values of support. This is also depicted in Figure 3.5, which shows the average discrepancy in confidence values according to the support. Clearly, rules with higher support values are more reliable.

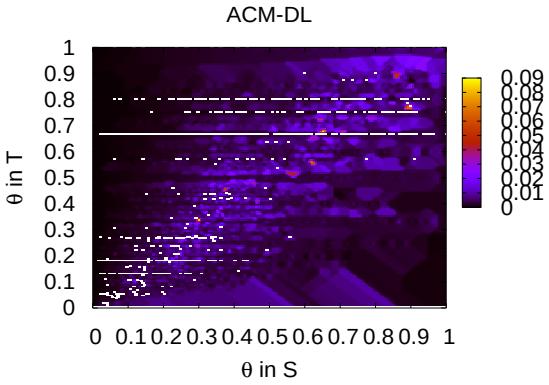

Figure 3.4. Rule confidence values in $\mathcal{S}$ and $\mathcal{T}$ as a function of rule support.

At first glance, one might expect that rules with lower values of support are not reliable, and should therefore be discarded. This is not always true, as shown in Figure 3.6, which shows the relationship between σ_{min}, execution time, and MicF_1. As it can be seen, higher MicF_1 numbers are obtained when lower σ_{min} values are employed. This is because some documents in the test set contain rare features, and consequently,

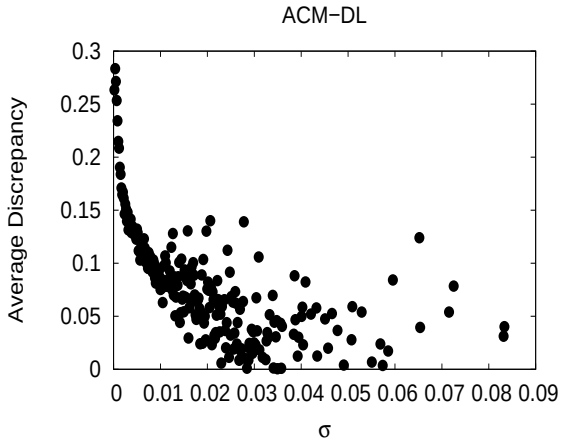

Figure 3.5. Average discrepancy of rule confidence as a function of rule support.

such documents may demand rules with low support values. Although the information provided by such rules may not be reliable, having this information is still better than having no information at all. The problem, however, is that the execution time increases exponentially as σ_{min} decreases. This happens because, an overwhelming number of decision rules are extracted from $\mathcal{S}$. This problem may prevent associative classification algorithms to achieve full potential. What is needed is a classification algorithm which is able to extract only indispensable rules, without incurring unnecessary overhead (even for lower values of σ_{min}). In the next chapter, we will present this algorithm.

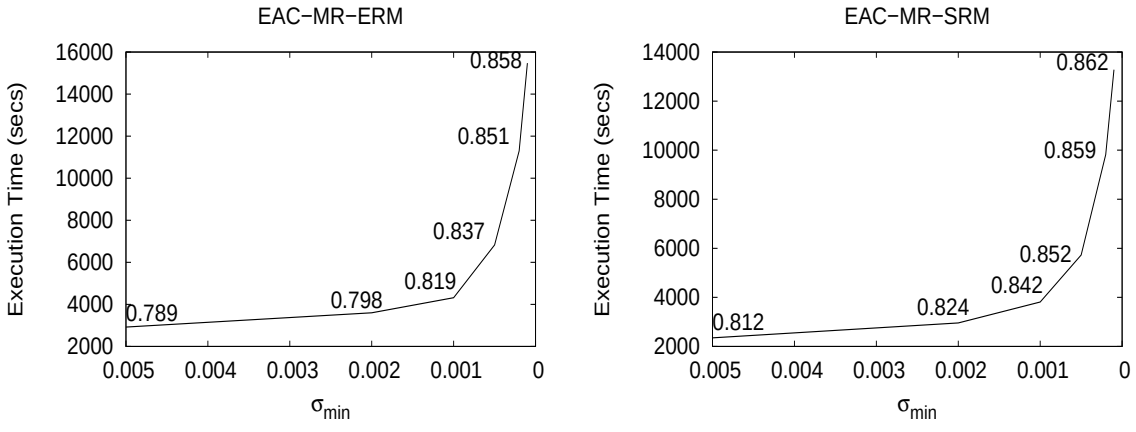

Figure 3.6. Relationship between σ_{min}, MicF$_1$, and execution time.

3.4 Related Work

Most existing work on associative classification relies on developing new algorithms to improve classification performance. The difference between these algorithms resides on the way they exploit decision rules. CBA [Liu et al., 1998] extracts decision rules from the training data, and ranks these rules according to their confidence. Ties are broken using their support. Then it selects the best ranked rule to be applied to each input in the test set. CBA was used as baseline in our experimental evaluation, and, on average, it showed to be superior than EAC-SR. On the other hand, EAC-MR, EAC-MR-ERM, and EAC-MR-SRM showed better classification performance than CBA. Enhancements to CBA were proposed in [Wang and Karypis, 2005; Li et al., 2001; Yin and Han, 2003; Dong et al., 1999]. HARMONY [Wang and Karypis, 2005] uses an input-centric rule-extraction approach in the sense that it assures the inclusion, in the final rule set, of at least one rule for each input in the training data. CMAR [Li et al., 2001] selects multiple rules (instead of a single best one), and the prediction is performed based on a weighted χ^2 analysis of the selected rules. An algorithm which uses only non-redundant decision rules for sake of prediction was proposed in [Baralis and Chiusano, 2004]. CAEP [Dong et al., 1999] exploits the concept of *emerging patterns* (patterns that present different frequencies in the training data, depending on the associated class), and, as a result, it usually predicts accurately all classes, even if their populations are unbalanced. Wang et al. [2000] proposed an algorithm that finds rules with high confidence, and then it prunes those ones that are not statistically meaningful using a decision tree like structure. It has been empirically shown that these associative classification algorithms usually outperform decision trees [Liu et al., 1998; Veloso et al., 2006b; Li et al., 2001].

Other approaches for rule-based classification, include algorithms such as RISE [Domingos, 1995], RIPPER [Cohen, 1995], and SLIPPER [Cohen and Singer, 1999]. The algorithms use greedy heuristics which are driven by global metrics. RISE performs a complete overfitting by considering each input as a rule, and then it generalizes the rules. RIPPER and SLIPPER extend the "overfit and prune" paradigm, that is, they start with a huge rule set and prune it using several heuristics. Further, the SLIPPER algorithm also associates a probability with each rule, weighting the contribution of the rule during prediction.

While EAC-SR and EAC-MR share some similarities with other associative classification algorithms (such as CBA and CMAR), EAC-MR-ERM and EAC-MR-SRM improve previous algorithms by introducing the use of empirical and structural risk minimization techniques in the context of associative classification. These improvements, as shown in our experiments, provide significant gains in classification performance.

3.5 Summary

In this chapter we have introduced associative classification algorithms. We showed that the basic algorithm, EAC-SR, is PAC-efficient. Although EAC-SR shares several similarities with other existing algorithms, we used it as a starting point for designing more efficient and practical algorithms. The first improvement to EAC-SR leads to EAC-MR, which produces a mapping function that employs multiple decision rules. Then, we apply function approximation strategies, such as empirical and structural risk minimization, in order to produce mapping function with appropriate complexities. These improvements lead to EAC-MR-ERM and EAC-MR-SRM algorithms. These algorithms were evaluated and compared to several baselines using different datasets. In the first set of experiments, using UCI benchmark datasets, the proposed improvements provided gains that range from 10.6% to 24.7%. Results obtained in the second set of experiments, using the ACM Digital Library, revealed some deficiencies that are inherent to EAC-based algorithms. Specifically, some inputs may demand decision rules that cannot be extracted in reasonable time by EAC-based algorithms. The solution for such deficiencies is the main focus of the next chapter.

Chapter 4

Demand-Driven Associative Classification

The ultimate goal of classification algorithms is to achieve the best possible classification performance for the problem at hand. Most often, classification performance is obtained by assessing some accuracy criterion using the test set, $\mathcal{T}$. Therefore, an effective classification algorithm does not need to approximate the target function over the entire input space (i.e., over all possible inputs). Rather, it needs only to approximate the parts of the target function that are defined over the inputs in $\mathcal{T}$.

As discussed, it is often hard to approximate the target function defined over inputs in $\mathcal{T}$, using a single mapping function. The key insight is to produce a specifically designed function, $f_{\mathcal{S}}^{x_i}$, which approximates the target function at each input $x_i \in \mathcal{T}$. Thus, a natural way to improve classification performance is to predict the outputs for inputs in $\mathcal{T}$, on a demand-driven basis. In this case, particular characteristics of each input in $\mathcal{T}$ may be taken into account while predicting the corresponding output. The expected result is a set of multiple mapping functions, where each function $f_{\mathcal{S}}^{x_i}$ is likely to perform particularly accurate predictions for input $x_i \in \mathcal{T}$, no matter the (possible poor) performance in predicting outputs for other inputs.

4.1 Method and Algorithms

In this section we will present a method for associative classification, which produces mapping functions on a demand-driven basis, according to each input in $\mathcal{T}$. First, it will be described how decision rules are extracted from $\mathcal{S}$, and then how these rules are used in order to approximate $\mathrm{P}(y|x)$. The method to be presented will hereafter be referred to as LAC (standing for Lazy Associative Classification). Algorithms based on this method will also be presented in this section.

Correlation Preserving Projection

In demand-driven associative classification, whenever an input $x_i \in \mathcal{T}$ is being considered, that input is used as a filter to remove from $\mathcal{S}$, features and examples that are useless[1] to approximate the target function at input x_i. This process generates a *projected training data*, defined as $\mathcal{S}^{x_i}$. Specifically, given an input $x_i \in \mathcal{T}$, the corresponding projected training data, $\mathcal{S}^{x_i}$, is the set of examples in $\mathcal{S}$, which is obtained after removing all features that are not included in x_i.

Theorem 4.1. *Given an input $x_i \in \mathcal{T}$, a decision rule $\mathcal{X} \rightarrow c_j$, such that $\mathcal{X} \subseteq x_i$, has the same confidence value in $\mathcal{S}$ and $\mathcal{S}^{x_i}$.*

Proof. Comes directly from the fact that confidence is null-invariant, thus adding or removing examples that do not contain $\mathcal{X}$, does not change $\theta(\mathcal{X} \rightarrow c_j)$. □

Theorem 4.1 states that the projection preserves the original correlation between inputs and outputs, since confidence values for rules extracted from $\mathcal{S}$ and $\mathcal{S}^{x_i}$ have the same value. Thus, the difference of extracting rules from $\mathcal{S}$ or from $\mathcal{S}^{x_i}$, resides basically on the rules that are used to approximate the target function at input x_i.

Demand-Driven Rule Extraction

Rule extraction is a major issue when devising an associative classification algorithm. Typically, frequent rules are extracted from $\mathcal{S}$, and used to produce a mapping function. Two problems may arise: (i) an infrequent, but useful rule may be not extracted from $\mathcal{S}$ (possibly hurting classification performance), and (ii) a frequent, but useless rule may be extracted from $\mathcal{S}$ (incurring unnecessary overhead). Figure 4.1 illustrates these problems. In the figure, black balls represent useful rules, and white balls represent useless rules. σ_{min} induces a border which separates frequent from infrequent rules. If σ_{min} is set too high, few useless rules are extracted from $\mathcal{S}$, but several useful rules are not extracted. If σ_{min} is set too low, all useful rules are extracted from $\mathcal{S}$, but a prohibitive number of useless rules is also extracted. An optimal value for σ_{min}, in the sense that only useful rules are extracted from $\mathcal{S}$, is unlikely to exist.

An ideal scenario would be to extract only useful rules from $\mathcal{S}$, without discarding useful ones. Inputs in $\mathcal{T}$ have valuable information that can be used during rule extraction to guide the search for useful rules. In this section, we propose to achieve the ideal scenario by extracting rules from $\mathcal{S}$ on a demand-driven basis, according to the input in $\mathcal{T}$ being considered.

[1] Usefulness is defined in Section 3.1.2.

Theorem 4.2. *Given an input $x_i \in \mathcal{T}$, $\mathcal{R}^{x_i}$ contains only decision rules that are useful to approximate the target function at x_i.*

Proof. Rules in $\mathcal{R}^{x_i}$ are extracted from $\mathcal{S}^{x_i}$. Since all inputs in $\mathcal{S}^{x_i}$ contain only features that are included in x_i, the existence of a rule $\mathcal{X} \rightarrow c_j \in \mathcal{R}^{x_i}$, such that $\mathcal{X} \not\subseteq x_i$, is impossible. □

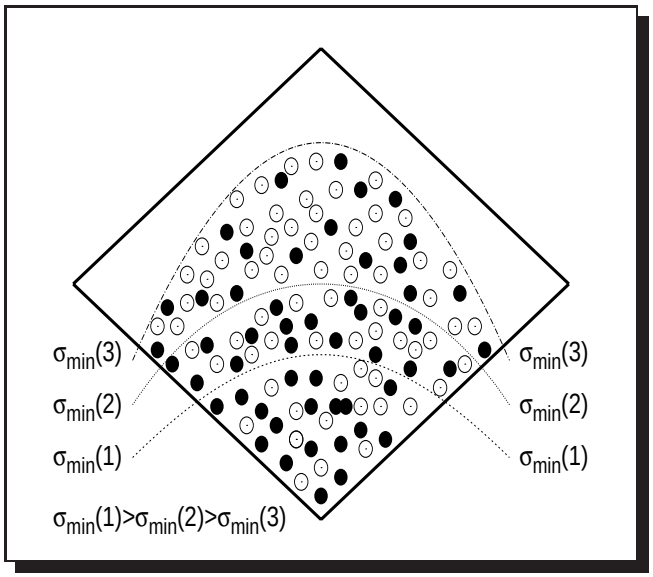

Figure 4.1. The pruning dilemma.

Thus, according to Theorem 4.2, only useful rules are extracted from $\mathcal{S}$. Next, we will discuss more sophisticated pruning strategies that reduce the chance of discarding useful rules.

Demand-Driven Pruning with Multiple Cut-Off Values

As discussed in the previous chapter, the typical pruning strategy is based on a cut-off frequency value derived from σ_{min}, which separates frequent from infrequent rules. In many cases, however, rules with low support may carry useful information for the sake of prediction. In such cases, the "use-and-abuse" support-based pruning strategy [Baralis et al., 2004] is not appropriate, since useful rules may also be discarded. We propose an alternate strategy, which prevents support-based pruning from being excessive. The proposed strategy employs multiple cut-off values. Specifically, cut-off values are calculated depending on how frequent (or how rare) are the features composing input $x_i \in \mathcal{T}$. The key insight is that if x_i contains commonly-appearing features, then the corresponding projected training data, $\mathcal{S}^{x_i}$, will contain several examples (i.e., $|\mathcal{S}^{x_i}| \approx |\mathcal{S}|$). Otherwise, if x_i contains rare features, then the corresponding projected

training data will contain only few examples (i.e., $|\mathcal{S}^{x_i}| \ll |\mathcal{S}|$). Therefore, for a fixed σ_{min}, the cut-off value for input x_i (which is denoted as $\pi^{x_i}_{min}$) is calculated based on the size (i.e., the number of examples) of $\mathcal{S}^{x_i}$, according to Equation 4.1.

$$\pi^{x_i}_{min} = \lceil \sigma_{min} \times |\mathcal{S}^{x_i}| \rceil \tag{4.1}$$

Since the cut-off value is calculated based on how frequent (or how rare) are the features composing input x_i, the chance of discarding rules matching x_i is reduced. This is because inputs $x_i \in \mathcal{T}$ composed of rare features will produce small projections of the training data, decreasing the value of $\pi^{x_i}_{min}$.

Caching Common Decision Rules

Extracting a decision rule has a significant computational cost, since it involves accessing $\mathcal{S}$ multiple times. Different inputs in $\mathcal{T}$ may demand the extraction of different rules, but it is very likely that some of these rules are common. In this case, caching (or memorization) is very effective in reducing work replication and, consequently, to reduce the number of accesses to $\mathcal{S}$.

Our cache is a pool of entries, and each entry has the form <*key*, *data*>, where key=$\{\mathcal{X}, c_j\}$ and $data$=$\{\sigma(\mathcal{X} \rightarrow c_j), \theta(\mathcal{X} \rightarrow c_j)\}$. Our cache implementation has a limited storage and stores all cached decision rules in main memory. Before extracting rule $\mathcal{X} \rightarrow c_j$, algorithms based on the LAC method first check whether this rule is already in the cache. If an entry is found with a key matching $\{\mathcal{X}, c_j\}$, then the rule in the cache entry is used instead of extracting it from $\mathcal{S}$. If it is not found, the rule is extracted from $\mathcal{S}$ and then it can be inserted into the cache.

The cache size is limited, and when the cache is full, some rules must be discarded to make room for other ones. The replacement heuristic is based on the support of the rules. More specifically, the least frequent rule stored in the cache is the first one to be discarded (and it will only be discarded if the rule to be inserted is more frequent than it). There are two main reasons to adopt this heuristic. First, the more frequent is the rule, the higher is the chance of using this rule later, to predict the output for other inputs in $\mathcal{T}$ (thus, if a frequent rule is discarded, it is higher the chance of recalculating it later on). Second, the computational cost associated with the extraction of more frequent rules is higher than the cost associated with the extraction of less frequent ones (more frequent rules necessitates more accesses to $\mathcal{S}$).

4.1.1 Prediction

Mapping functions perform predictions using the decision rules extracted from $\mathcal{S}^{x_i}$, as will be discussed next.

LAC-SR – This algorithm employs a single decision rule, which is the strongest one in $\mathcal{R}^{x_i}$ (i.e., the one with highest confidence value), in order to predict the output for input $x_i \in \mathcal{T}$, according to Equation 4.2.

$$f_{\mathcal{S}}^{x_i}(x_i) = c_j \text{ such that } \theta(\mathcal{X} \rightarrow c_j) \text{ is argmax}(\theta(r)) \; \forall \; r \in \mathcal{R}^{x_i} \tag{4.2}$$

LAC-MR – This algorithm employs multiple decision rules to produce a specific function, $f_{\mathcal{S}}^{x_i}$, for input $x_i \in \mathcal{T}$. It first projects the original training data, $\mathcal{S}$, according to input x_i. The result is $\mathcal{S}^{x_i}$. Then, LAC-MR extracts rules from $\mathcal{S}^{x_i}$, on a demand-driven basis. All extracted rules are useful to x_i. Basic steps for LAC-MR are shown in Algorithm 5.

Algorithm 5 Finding $f_{\mathcal{S}}^{x_i}$, according to LAC-MR.

Require: The training data $\mathcal{S}$, input $x_i \in \mathcal{T}$, $\sigma_{min}, and \text{lf}_{\mathcal{S}}^{x_i}$.

Ensure: $\mathcal{S}^{x_i} \Leftarrow \mathcal{S}$ projected according to x_i
2: $\mathcal{R}^{x_i} \Leftarrow$ rules $\mathcal{X} \rightarrow c_j$ extracted from $\mathcal{S}^{x_i}$, such that $\pi(\mathcal{X} \rightarrow c_j) \geq \pi_{min}^{x_i}$
3: **r**eturn $f_{\mathcal{S}}^{x_i}$ such that $f_{\mathcal{S}}^{x_i}(x_i) = c_j$, where $\hat{p}(c_j|x_i)$ is argmax$(\hat{p}(c_k|x_i)) \; \forall \; 1 \leq k \leq p$

		Input (x_i)			Output (y_i)
		a_1	a_2	a_3	
$\mathcal{S}$	(x_1, y_1)	[0.00-0.22]	[0.33-0.71]	[0.00-0.35]	1
	(x_2, y_2)	[0.00-0.22]	[0.33-0.71]	[0.35-1.00]	0
	(x_3, y_3)	[0.46-1.00]	[0.71-1.00]	[0.35-1.00]	1
	(x_4, y_4)	[0.22-0.46]	[0.00-0.33]	[0.35-1.00]	0
	(x_5, y_5)	[0.00-0.22]	[0.33-0.71]	[0.00-0.35]	1
	(x_6, y_6)	[0.22-0.46]	[0.33-0.71]	[0.00-0.35]	1
	(x_7, y_7)	[0.00-0.22]	[0.33-0.71]	[0.00-0.35]	1
	(x_8, y_8)	[0.22-0.46]	[0.71-1.00]	[0.00-0.35]	1
	(x_9, y_9)	[0.46-1.00]	[0.00-0.33]	[0.35-1.00]	1
	(x_{10}, y_{10})	[0.22-0.46]	[0.00-0.33]	[0.35-1.00]	0
$\mathcal{T}$	(x_{12}, y_{12})	[0.46-1.00]	[0.00-0.33]	[0.35-1.00]	?(1)

Table 4.1. Training data and test set given as example.

Example Consider the example shown in Table 4.1. There are 10 pairs in $\mathcal{S}$. There is one input in $\mathcal{T}$, x_{12}, for which the corresponding output, y_{12}, is unknown. All inputs were discretized. After projecting $\mathcal{S}$ according to input x_{12}, we obtain $\mathcal{S}^{x_{12}}$, which is shown in Table 4.2.

		Input (x_i)			Output (y_i)
		a_1	a_2	a_3	
$\mathcal{S}^{x_{12}}$	(x_2, y_2)			[0.35-1.00]	0
	(x_3, y_3)	[0.46-1.00]		[0.35-1.00]	1
	(x_4, y_4)		[0.00-0.33]	[0.35-1.00]	0
	(x_9, y_9)	[0.46-1.00]	[0.00-0.33]	[0.35-1.00]	1
	(x_{10}, y_{10})		[0.00-0.33]	[0.35-1.00]	0

Table 4.2. Projected training data: $\mathcal{S}^{x_{12}}$.

Suppose σ_{min} is set to 0.30. In this case, $\mathcal{R}^{x_{12}}$ contains the following 6 rules:

1. $\{a_1 =$[0.46-1.00]$\rightarrow output = 1\}$ $(\theta = 1.00)$
2. $\{a_1 =$[0.46-1.00]$\wedge a_3 =$[0.35-1.00]$\rightarrow output = 1\}$ $(\theta = 1.00)$
3. $\{a_2 =$[0.00-0.33]$\rightarrow output = 0\}$ $(\theta = 0.67)$
4. $\{a_2 =$[0.00-0.33]$\wedge a_3 =$[0.35-1.00]$\rightarrow output = 0\}$ $(\theta = 0.67)$
5. $\{a_3 =$[0.35-1.00]$\rightarrow output = 0\}$ $(\theta = 0.60)$
6. $\{a_3 =$[0.35-1.00]$\rightarrow output = 1\}$ $(\theta = 0.40)$

According to Equation 3.6, LAC-MR calculates the likelihood associated with each output, which are $\hat{p}(\text{output} = 0|x_{12})$=0.48 and $\hat{p}(\text{output} = 1|x_{12})$=0.52. Thus. LAC-MR predicts output 1 for input x_{12}, which is the correct one.

4.1.2 Demand-Driven Function Approximation

A complex target function may be composed of simple parts. Thus, instead of approximating a complex target function using a complex mapping function (i.e., $f_{\mathcal{S}}$), we can employ multiple simple functions (i.e., $f_{\mathcal{S}}^{x_i}$). Intuitively, such simple functions are more likely to generalize than a single complex function. The appropriate complexity for each function $f_{\mathcal{S}}^{x_i}$ is selected using the function approximation techniques described in Section 2.3. LAC-MR-ERM employs the well known Empirical Risk Minimization technique to approximate the target function, while LAC-MR-SRM employs the Structural Risk Minimization technique to approximate the target function.

LAC-MR-ERM and LAC-MR-SRM are much finer-grained than their eager counterparts EAC-MR-ERM and EAC-MR-SRM, since they are able to select a different complexity for each function $f_{\mathcal{S}}^{x_i}$. In the end of the process these algorithms produce multiple mapping functions, each one with a possibly different complexity. Both algorithms will be described next.

LAC-MR-ERM – The hypothesis space induced by each projected training data, $\mathcal{S}^{x_i}$, is traversed by evaluating candidate functions in increasing order of complexity. That is, simpler functions are produced before more complex ones. To do this, we introduce a nested structure of subsets of rules, $h_1, h_2, \ldots, h_l$, where h_i contains frequent decision rules $\mathcal{X} \rightarrow c_j$ for which $|\mathcal{X}| \leq l$. Clearly, $h_1 \subseteq h_2 \subseteq \ldots \subseteq h_l$. The simplest candidate function is the one which uses decision rules in h_1 (i.e., this function uses only decision rules of size 1), while the most complex one uses decision rules in h_l. From these candidate functions, we select the one which minimizes Equation 2.9. This selection process involves a trade-off: as the complexity of the rules increases, the empirical error decreases, but the stability of the corresponding function also decreases. The selected function is the one that best trades empirical error and stability. Algorithm 6 shows the basic steps of LAC-MR-ERM.

Algorithm 6 Finding $f_{\mathcal{S}}^{x_i}$, according to LAC-MR-ERM.

Require: The training data $\mathcal{S}$, input $x_i \in \mathcal{T}$, σ_{min}, l, and δ
Ensure: $f_{\mathcal{S}}^{x_i}$.

1: *tighest bound* $\Leftarrow \infty$
2: $\mathcal{S}^{x_i} \Leftarrow \mathcal{S}$ projected according to x_i
3: **for** $i = 1$ to l **do**
4: $\quad \mathcal{R}^{x_i} \Leftarrow h_i \Leftarrow$rules $\mathcal{X} \rightarrow c_j$ extracted from $\mathcal{S}^{x_i}$, such that $|\mathcal{X}| \leq i$ and $\pi(\mathcal{X} \rightarrow c_j) \geq \pi_{min}^{x_i}$
5: $\quad$ **for** each pair $s_i = (x_i, y_i) \in \mathcal{S}^{x_i}$ **do**
6: $\quad\quad \beta_{s_i} = |f_{\mathcal{S}}^{x_i}(x_i) - f_{\mathcal{S}-s_i}^{x_i}(x_i)|$
7: $\quad$ **end for**
8: $\quad \beta = \sup(\beta_{s_i})$
9: $\quad bound \Leftarrow \mathcal{I}_{\mathcal{S}^{x_i}}[f_{\mathcal{S}}^{x_i}] + \left(\beta + (4|\mathcal{S}^{x_i}|\beta + 1) \times \sqrt{\frac{\ln \frac{1}{\delta}}{2|\mathcal{S}^{x_i}|}}\right)$
10: $\quad$ **if** *tighest bound* $\leq$ *bound* **then** break
11: $\quad$ *tighest bound* $\Leftarrow$ *bound*
12: **end for**
13: **return** $f_{\mathcal{S}}^{x_i}$ such that $f_{\mathcal{S}}^{x_i}(x_i) = c_j$, where $\hat{p}(c_j|x_i)$ is $\text{argmax}(\hat{p}(c_k|x_i))\ \forall\ 1 \leq k \leq p$

LAC-MR-SRM – The hypothesis space induced by each projected training data, $\mathcal{S}^{x_i}$, is traversed by evaluating candidate functions in increasing order of complexity.

That is, simpler functions are produced before more complex ones. Again, it is employed a nested structure of subsets of rules, $h_1, h_2, \ldots, h_l$, where h_i contains frequent decision rules $\mathcal{X} \rightarrow c_j$ for which $|\mathcal{X}| \leq l$. From these candidate functions, the selected is the one which minimizes Equation 2.10. This selection process involves a trade-off: as the complexity of the rules increases the empirical error decreases, but the VC-dimension of the corresponding function increases. The selected function is the one that best trades empirical error and VC-dimension. Algorithm 7 shows the detailed steps of LAC-MR-SRM.

Algorithm 7 Finding $f_{\mathcal{S}}^{x_i}$, according to LAC-MR-SRM.

Require: The training data $\mathcal{S}$, input $x_i \in \mathcal{T}$, σ_{min}, and δ
Ensure: $f_{\mathcal{S}}^{x_i}$.

1: *tighest bound* $\Leftarrow \infty$
2: $\mathcal{S}^{x_i} \Leftarrow \mathcal{S}$ projected according to x_i
3: **for** $i = 1$ to l **do**
4: $\quad \mathcal{R}^{x_i} \Leftarrow h_i \Leftarrow$rules $\mathcal{X} \rightarrow c_j$ extracted from $\mathcal{S}^{x_i}$, such that $|\mathcal{X}| \leq i$ and $\pi(\mathcal{X} \rightarrow c_j) \geq \pi_{min}^{x_i}$
5: $\quad d_{f_{\mathcal{S}}^{x_i}} = |\mathcal{S}^{x_i}| \times (1 - \mathcal{I}_{\mathcal{S}^{x_i}}[f_{\mathcal{S}}^{x_i}])$
6: $\quad bound \Leftarrow \mathcal{I}_{\mathcal{S}^{x_i}}[f_{\mathcal{S}}^{x_i}] + \sqrt{\frac{d_{f_{\mathcal{S}}^{x_i}}\left(\ln \frac{2|\mathcal{S}^{x_i}|}{d_{f_{\mathcal{S}}^{x_i}}} + 1\right) - \ln \frac{\delta}{4}}{|\mathcal{S}^{x_i}|}}$
7: $\quad$ **if** *tighest bound* $\leq$ *bound* **then** break
8: $\quad tighest\ bound \Leftarrow bound$
9: **end for**
10: **return** $f_{\mathcal{S}}^{x_i}$ such that $f_{\mathcal{S}}^{x_i}(x_i) = c_j$, where $\hat{p}(c_j|x_i)$ is argmax$(\hat{p}(c_k|x_i))$ $\forall\ 1 \leq k \leq p$

4.2 Empirical Results

In this section we will present the experimental results for the evaluation of the proposed demand-driven associative classification algorithms, which include: LAC-SR, LAC-MR, LAC-MR-ERM, LAC-MR-SRM.

Setup Continuous attributes were discretized using the MDL-based entropy minimization method [Fayyad and Irani, 1993], which was described in Section 3.1.1. In all the experiments we used 10-fold cross-validation and the final results of each experiment represent the average of the ten runs. All the results to be presented were found statistically significant based on a t-test at 95% confidence level.

4.2.1 The UCI Benchmark

Baselines The evaluation is based on a comparison involving a lazy decision tree algorithm (LazyDT), a kNN algorithm (kNN), and an associative classification algorithm (DeEPs). For LazyDT and kNN, we used the corresponding implementations available in MLC++ [Kohavi et al., 1996]. For DeEPs, we used the results available at [Li et al., 2004].

Parameters For LazyDT, we used the C4.5-auto-parm tool, available in MLC++, for finding optimum parameters for each dataset. For kNN, the value of k was carefully hand-tuned for each dataset. For LAC-SR, LAC-MR, LAC-MR-ERM, and LAC-MR-SRM, we set σ_{min}=0.01.

Evaluation Criteria Classification performance is expressed using the conventional true error rate in the test set.

Analysis Table 4.3 shows classification performance for different classification algorithms. Best results, including statistical ties, are shown in bold. LAC-SR outperforms LazyDT and kNN. In fact, it was demonstrated in [Veloso et al., 2006b] that, if we set all algorithms under the information gain principle, a demand-driven associative classification algorithm always outperforms the corresponding decision tree one. LAC-SR has also outperformed EAC-SR, providing gains of more than 8%. LAC-MR outperforms all baselines. It also outperformed EAC-MR, with gains of more than 7%. Furthermore, LAC-MR-ERM and LAC-MR-SRM were the best performers, suggesting that more finer-grained function approximation strategies are effective in improving classification performance. LAC-MR-ERM outperformed EAC-MR-ERM with gains of more than 9.5%, while LAC-MR-SRM outperformed EAC-MR-SRM with gains of more than 9.7%.

4.2.2 The ACM Digital Library

Baselines The evaluation is based on a comparison involving general-purpose algorithms, such as kNN, SVM, and TSVM (transductive SVM). For kNN, we used the implementation available in MLC++ [Kohavi et al., 1996]. For SVM, we used the implementation available at `http://svmlight.joachims.org/` (version 3.0). For TSVM, we used the implementation available at `http://www.kyb.mpg.de/bs/people/fabee/universvm.html`. Application-specific methods were also used in the evaluation. Amsler [Amsler, 1972] is a very used bibliographic-based method for document categorization. The Bayesian method

	LAC				Baselines		
Dataset	SR	MR	MR-ERM	MR-SRM	LazyDT	kNN	DeEPs
anneal	**0.025**	**0.025**	**0.021**	**0.021**	0.042	0.100	0.050
australian	0.146	0.138	0.132	0.132	0.152	0.152	**0.116**
auto	0.176	**0.156**	**0.151**	**0.151**	0.247	0.280	0.273
breast	0.074	**0.024**	**0.021**	**0.021**	0.051	0.185	0.036
cleve	0.142	**0.132**	**0.132**	**0.129**	0.172	0.162	0.158
crx	**0.127**	**0.127**	**0.123**	**0.123**	0.169	0.169	**0.119**
diabetes	**0.221**	**0.221**	**0.221**	**0.221**	0.249	0.241	0.230
german	0.265	**0.256**	**0.247**	**0.247**	0.261	**0.256**	**0.256**
glass	0.253	0.253	**0.231**	0.243	0.265	0.372	0.326
heart	0.168	0.144	0.144	0.137	0.177	**0.102**	0.177
hepatitis	0.183	0.183	**0.110**	**0.110**	0.203	0.223	0.175
horse	0.176	0.176	0.173	0.173	0.173	0.173	**0.147**
hypo	0.062	0.047	0.032	0.032	**0.012**	**0.012**	**0.018**
ionosphere	0.077	**0.063**	**0.063**	**0.063**	0.080	0.153	0.088
iris	0.053	**0.033**	**0.033**	**0.033**	0.053	0.044	**0.033**
labor	0.033	0.033	0.033	**0.016**	0.204	0.033	0.023
led7	0.281	**0.259**	0.271	**0.263**	**0.265**	**0.263**	**0.263**
lymph	0.180	**0.130**	**0.130**	**0.123**	0.201	0.180	0.246
pima	0.266	**0.218**	**0.208**	**0.212**	0.259	0.279	0.229
sick	0.061	0.061	0.061	**0.026**	**0.021**	0.094	**0.033**
sonar	0.220	**0.067**	**0.067**	**0.067**	0.246	0.220	0.133
tic-tac-toe	0.054	0.036	0.036	0.054	**0.006**	0.108	**0.004**
vehicle	0.310	0.310	0.292	0.292	0.318	0.334	**0.254**
waveform	0.225	0.225	0.212	0.196	0.225	0.225	**0.157**
wine	0.050	**0.000**	**0.000**	**0.000**	0.079	0.263	0.039
zoo	0.096	0.096	0.070	0.070	0.078	0.070	**0.028**
Avg	0.149	0.132	**0.124**	**0.121**	0.162	0.180	0.139

Table 4.3. Classification performance of different algorithms.

[Calado et al., 2003] and Multi-Kernel SVMs [Joachims et al., 2001] are two state-of-the-art representatives of methods for document categorization.

Parameters The implementation for Amsler and Bayesian methods are non-parametric. For SVM, TSVM, and Multi-Kernel SVMs, we used the grid parameter search tool in LibSVM [Chang and Lin, 2001] for finding the optimum parameters. For kNN, we carefully tuned k=15. For LAC-MR, LAC-MR-ERM, and LAC-MR-SRM, we set σ_{min}=0.005.

Evaluation Criteria Categorization performance for the various methods being evaluated, is expressed through F_1 measures. In this case, precision p is defined as

the proportion of correctly classified documents in the set of all documents. Recall r is defined as the proportion of correctly classified documents out of all the documents having the target category. F_1 is a combination of precision and recall defined as the harmonic mean $\frac{2pr}{p+r}$. Macro- and micro-averaging [Yang et al., 2002] were applied to F_1 to get single performance values over all classification tasks. For F_1 macro-averaging ($MacF_1$), scores were first computed for individual categories and then averaged over all categories. For F_1 micro-averaging ($MicF_1$), the decisions for all categories were counted in a joint pool. The computational efficiency is evaluated through the total execution time, that is, the processing time spent in training and classifying all documents.

Analysis Table 4.4 shows categorization performance for various classification algorithms. Again, Amsler was used as the baseline for comparison. The TSVM and Multi-Kernel algorithms achieved the highest values of $MicF_1$ and $MicF_1$. LAC-MR outperformed these algorithms, showing the importance of producing mapping functions on a demand-driven basis. Finer-grained function approximation techniques were also effective. LAC-MR-ERM and LAC-MR-SRM were the best overall performers. Further, demand-driven associative classification algorithms are much faster than most of the baselines.

Algorithms	$MicF_1$	$MacF_1$	Gains (%) over baseline		Execution Time
			$MicF_1$	$MacF_1$	
Amsler (baseline)	0.832	0.783	–	–	1,251 secs
kNN	0.833	0.774	0.001	-0.011	83 secs
SVM	0.845	0.810	0.016	0.035	1,932 secs
Bayesian	0.847	0.796	0.019	0.016	8,281 secs
TSVM	0.855	0.808	0.028	0.032	17,183 secs
Multi-Kernel	0.859	0.812	0.032	0.037	14,894 secs
LAC-MR	0.862	0.814	0.036	0.040	257 secs
LAC-MR-ERM	0.868	0.833	0.043	0.064	504 secs
LAC-MR-SRM	0.873	0.839	0.049	0.071	342 secs

Table 4.4. Categorization performance for different algorithms.

Figure 4.2 depicts the execution times obtained by employing different cache sizes. We varied the cache size from 0 to 50MB, and for each storage capacity we obtained the corresponding execution time. Clearly, execution time is very sensitive to cache size. Caches as large as 50MB are able to store all decision rules with no need of replacement, being the best cache configuration.

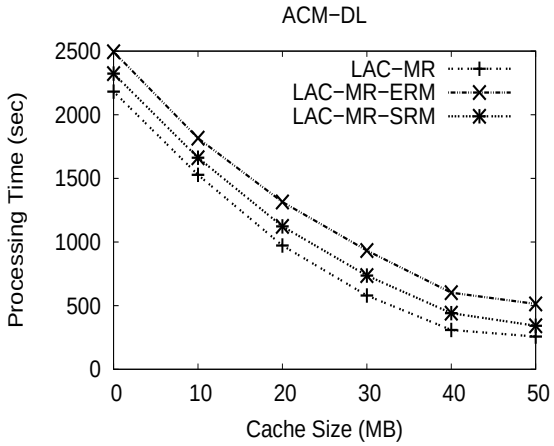

Figure 4.2. Processing time with varying cache sizes..

4.3 Related Work

We believe that lazy classification algorithms are the closest to demand-driven classification algorithms. Most existing work on lazy [Stanfil and Waltz, 1986; Aha, 1997] (memory-based, instance-based, case-based, local-learning) classification is based on the kNN approach [Dasarathy, 1990; Bottou and Vapnik, 1992; Yang, 1994], which retrieves a set of similar related instances at classification time, and then uses the retrieved instances to build a local approximation of the target function. It was showed in [Aha et al., 1991] that storing and using specific instances improves the performance of several classification algorithms, including algorithms that learn decision trees [Friedman et al., 1996; Fern and Brodley, 2003] and Bayesian rules [Zheng and Webb, 2000; Kontkanen et al., 1998]. In [Friedman et al., 1996] a lazy learning algorithm for decision tree induction was proposed. This algorithm tries to alleviate the missing rule problem (i.e., when no path in the tree matches an input in $\mathcal{T}$), since a specific decision tree is induced for each input in $\mathcal{T}$.

Our work shares some similarities with [Li et al., 2004], in the sense that they sample the training set at classification time, and use a similar language to express the induced concept. However, the algorithm proposed in [Li et al., 2004] uses emerging patterns as basic seeds for the rules that will compose the concept. The use of emerging patterns is a tentative to capture important, but possibly not frequent, rules. On the other hand, our algorithms attempt to capture important, but not so frequent rules, by employing different cut-off values while traversing the search space for rules on a demand-driven basis.

Transductive classification algorithms also share some similarities with demand-driven classification algorithms. Transductive support vector machines [Vapnik, 1995]

are a method of improving the effectiveness of SVMs [Boser et al., 1992] by using inputs in $\mathcal{T}$. Transductive SVMs, like regular SVMs, learn a large margin hyperplane function using examples in $\mathcal{S}$, but simultaneously force this hyperplane to be far away from the inputs in $\mathcal{T}$. Vapnik [1995] interpreted the success of transductive SVMs due to the fact that transduction (labeling a specific test set) is inherently easier than induction (learning a general rule from the training data). Unfortunately, transductive SVM algorithms are often unable to deal with a large number of examples. The first implementation of these algorithms appeared in [Bennett and Demiriz, 1998a], which used an integer programming method, intractable for large classification problems. Joachims [1999] then proposed a combinatorial approach, known as SVMLight-TSVM, that is practical for a few thousand examples. Fung and Mangasarian [2001] introduced a sequential optimization procedure that could potentially scale well, although their largest experiment used only 1,000 examples. The most practical transductive SVM algorithm seems to be univerSVM, which was proposed in [Collobert et al., 2006]. Demand-driven associative classification algorithms exploit inputs in $\mathcal{T}$ in a different way. Specifically, inputs in $\mathcal{T}$ are used to reduce the number of decision rules that are extracted from $\mathcal{S}$ and to select mapping functions with appropriate complexities. In our experiments (including experiments presented in Chapter 5), we show that our demand-driven classification algorithms scale well for large scale problems.

4.4 Summary

In this chapter we have introduced demand-driven associative classification algorithms. These algorithms postpone the rule extraction process, so that rules are only extracted from the training data after an input in the test set is informed. Each input in the test set induces a sub-problem, which is a projection of the original training data. Demand-driven algorithms have some important abilities. First, they are able to extract only useful decision rules from the training data. LAC-based algorithms exploit this ability to overcome most of the deficiencies of EAC-based algorithms. Further, LAC-based algorithms are also able to take into account specific characteristics of each sub-problem while producing mapping functions. They apply multiple minimum support thresholds, which are calculated on-the-fly, according to how frequent (or how rare) are the features that composed each sub-problem. A simple caching mechanism makes rule extraction fast. Further, LAC-MR-ERM and LAC-MR-SRM are also able to produce mapping functions with appropriate complexities for each input in the test set. Specificaly, certain inputs may induce simple sub-problems, which demand a simple mapping function. Other inputs may induce complex sub-problems, which demand

a complex mapping function. Our experiments have shown that LAC-based algorithms are competitive with the state-of-the-art, while being much faster. In the first set of experiments, using UCI benchmark datasets, LAC-based algorithms provided gains in terms of classification performance ranging from 12.9% to 32.8%. In the second set of experiments, LAC-based algorithms provided gains in terms of classification performance that are up to 4.9%, while being one order of magnitude faster than the most competitive baselines.

Chapter 5

Extensions to Demand-Driven Associative Classification

Some application scenarios are not well suited for the direct application of the classification algorithms presented so far. In this chapter we will discuss extensions for demand-driven associative classification, so that its spectrum of applications is enlarged. Such extensions include: multi-label classification, multi-metric classification, calibrated classification, self-training, and ranking.

5.1 Multi-Label Classification

A typical assumption in classification is that outputs are mutually exclusive, so that an input can be mapped to only one output (i.e., single-label classification). However, due to ambiguity or multiplicity, it is quite natural that many applications violate this assumption, allowing inputs to mapped to multiple outputs simultaneously. Multi-label classification is a generalization of single-label classification, and its generality makes it much more difficult to solve.

Despite its importance, research on multi-label classification is still lacking. Common approaches simply learn independent functions, not exploiting dependencies among outputs. Also, several small disjuncts may appear due to the possibly large number of combinations of outputs, and neglecting these small disjuncts may degrade classification performance. In this section we extend demand-driven associative classification to multi-label classification. The proposed method progressively exploits dependencies among outputs.

5.1.1 Related Work

Typical algorithms for multi-label classification are based on producing an independent binary classification function for each output (or label). These independent functions are used to assign a probability of membership to each output, and then an instance is mapped to the outputs that rank above a given threshold. Examples of this approach include ADTBoost.MH [Comité et al., 2003] (decision trees that can directly handle multi-label problems), a multi-label generalization of SVM algorithms [Boser et al., 1992], and a a multi-label lazy learning based on the kNN method [Zhang and Zhou, 2007].

The main problem with the binary, independent approach is that it does not consider correlation among outputs. The direct multi-label approach explores this correlation by considering a combination of outputs as a new, separate label [Boutell et al., 2004]. For instance, a multi-label problem with 10 original outputs will be transformed to a single-label problem composed of potentially 1,024 possible outputs. The problem now is that a relatively small number of examples may be associated with those new outputs, specially for large combinations.

Research in multi-label classification was initially motivated by the difficulty encountered in text categorization tasks [Schapire and Singer, 2000] due to ambiguity. In fact, many multi-label classification algorithms are specific to text categorization applications. However, it is also important in other domains, such as pattern recognition and bioinformatics. In [Boutell et al., 2004] a multi-label algorithm was used for semantic scene classification. In [Clare and King, 2001] the C4.5 Decision Tree algorithm was adapted to handle multiple outputs, and used in gene expression tasks.

5.1.2 Algorithms

Next we present two algorithms for multi-label associative classification. The first one, which will be referred to as LAC-MR-IO (standing for LAC-MR with independent outputs), neglects any association among outputs while producing classification functions. The second one, which will be hereafter referred to as LAC-MR-CO (standing for LAC-MR with correlated outputs), explores correlated outputs, improving classification performance.

LAC-MR-IO – The assumption of independence of outputs leads to a natural strategy for multi-label classification, where class membership probabilities, $\hat{p}(c_i|x)$, are naturally used to select outputs. A user specified threshold, Δ_{min} ($0 \leq \Delta_{min} \leq 0.5$), is used to separate the outputs that will be predicted. Specifically, for a given input

x, an output c_i is only predicted if $\hat{p}(c_i|x) \geq \Delta_{min}$. LAC-MR-IO follows this strategy, and the basic steps are shown in Algorithm 8.

Algorithm 8 Finding $f_{\mathcal{S}}^{x_i}$, according to LAC-MR-IO.

Require: The training data $\mathcal{S}$, input $x_i \in \mathcal{T}$, σ_{min}, and Δ_{min}
Ensure: $f_{\mathcal{S}}^{x_i}$.

1: $\mathcal{S}^{x_i} \Leftarrow \mathcal{S}$ projected according to x_i
2: $\mathcal{R}^{x_i} \Leftarrow$ rules $\mathcal{X} \rightarrow c_j$ extracted from $\mathcal{S}^{x_i}$, such that $\pi(\mathcal{X} \rightarrow c_j) \geq \sigma_{min} \times |\mathcal{S}^{x_i}|$
3: **return** $f_{\mathcal{S}}^{x_i}$=$\mathcal{P}^{x_i}$, such that $\forall c_j \in \mathcal{P}^{x_i}$, $\hat{p}(c_j|x_i) \geq \Delta_{min}$

LAC-MR-CO – Outputs in multi-label problems are often correlated, and as it will be shown in the experiments, this correlation may be helpful for improving classification performance. Next we will present LAC-MR-CO, which, unlike LAC-MR-IO, explicitly exploits interactions among outputs while producing classification functions. These functions are composed of *multi-label decision rules*, which are rules of the form $\{\mathcal{X} \cup \mathcal{F}\} \rightarrow c_j$, where $\mathcal{F} \subseteq \{\mathcal{C} - c_j\}$. Thus, multi-label decision rules enable the presence of outputs in the antecedent.

Classification functions are produced iteratively, following a greedy heuristic called *progressive label focusing* [Veloso et al., 2007a], which tries to find the best combination of outputs by making locally best choices. In the first iteration, $\mathcal{F}$=$\emptyset$, and a set of rules matching input $x \in \mathcal{T}$, $\mathcal{R}^{x^1}$ (which is composed of rules of the form $\mathcal{X} \rightarrow c_j$), is extracted from $\mathcal{S}^x$. Based on $\mathcal{R}^{x^1}$, output c_r is predicted for input x. In the second iteration, output c_r is treated as a new feature while extracting rules (i.e., $\mathcal{F}$=$\{c_r\}$). A set of multi-label rules, $\mathcal{R}^{x^2}$ (which is composed of rules of the form $\{\mathcal{X} \cup \{c_r\}\} \rightarrow c_j$, with $j \neq r$), is extracted from $\mathcal{S}^x$. Based on $\mathcal{R}^{x^2}$, output c_s is predicted for input x. This process iterates until no more rules are extracted from $\mathcal{S}^x$. The basic idea is to progressively narrow the search space for rules as outputs are being predicted for input x. The main steps of LAC-MR-CO are shown in Algorithm 9.

Example – Table 5.1 shows an example of the multi-label classification problem. In this case, each input corresponds to a movie, and each movie is assigned to one or more labels (i.e., outputs). This movie subject was chosen because its intuitive aspects may help to understand the ideas just discussed.

Suppose we want to predict the outputs for input x_{11}. In this case, the execution of LAC-MR-IO, with σ_{min}=0.50 and Δ_{min}=0.35, proceeds as follows. First a set of frequent rules is extracted from $\mathcal{S}^{x_{11}}$:

1. actor=M. Damon →output=Action (θ=1.00)

Algorithm 9 Finding $f_{\mathcal{S}}^{x_i}$, according to LAC-MR-CO.

Require: The training data $\mathcal{S}$, input $x_i \in \mathcal{T}$, and σ_{min}
Ensure: $f_{\mathcal{S}}^{x_i}$.

1: $\Omega \Leftarrow 1$
2: $\mathcal{F} \Leftarrow \emptyset$
3: $\mathcal{S}^{x_i} \Leftarrow \mathcal{S}$ projected according to x_i
4: **while** true **do**
5: $\quad \mathcal{R}^{x^i} \Leftarrow$ rules $\{\mathcal{X} \cup \mathcal{F}\} \rightarrow c_j$ extracted from $\mathcal{S}^{x_i}$, such that $\pi(\{\mathcal{X} \cup \mathcal{F}\} \rightarrow c_j) \geq \sigma_{min} \times |\mathcal{S}^{x_i}|$
6: $\quad$ **if** $\mathcal{R}^{x^i}=\emptyset$ **then** break
7: $\quad \mathcal{F} \Leftarrow \mathcal{F} \cup c^{\Omega}$, where $\hat{p}(c^{\Omega}|x_i)$ is argmax$(\hat{p}(c_k|x_i))$ with $1 \leq k \leq p$
8: $\quad \Omega{+}{+}$
9: **end while**
10: **return** $f_{\mathcal{S}}^{x_i}=\{c^1, \ldots, c^{\Omega}\}$

		Output	Input	
			Title	Actors
$\mathcal{S}$	x_1	Comedy/Romance	Forrest Gump	T. Hanks
	x_2	Drama/Romance	The Terminal	T. Hanks
	x_3	Drama/Crime	Catch Me If You Can	T. Hanks and L. DiCaprio
	x_4	Drama/Crime	The Da Vinci Code	T. Hanks
	x_5	Drama/Crime	Blood Diamond	L. DiCaprio
	x_6	Crime/Action	The Departed	L. DiCaprio and M. Damon
	x_7	Crime/Action	The Bourne Identity	M. Damon
	x_8	Action/Romance	Syriana	M. Damon
	x_9	Romance	Troy	B. Pitt
	x_{10}	Drama/Crime	Confidence	E. Burns
$\mathcal{T}$	x_{11}	[Action/Crime]	Ocean's Twelve	B. Pitt and M. Damon
	x_{12}	[Crime/Drama]	The Green Mile	T. Hanks

Table 5.1. Training data given as example of a multi-label problem.

2. actor=M. Damon →output=Crime (θ=0.67)

Following Equation 3.5, $\hat{p}$(Action$|x_{11}$)=0.60, and $\hat{p}$(Crime$|x_{11}$)=0.40. In this case, both outputs, "Action" and "Crime", are correctly predicted for input x_{11}, since Δ_{min}=0.35.

Now, suppose we want to predict the outputs for input x_{12}. In this case, the execution of LAC-MR-CO, with σ_{min}=0.50, proceeds as follows. At the first iteration, one rule is extracted from $\mathcal{S}^{x_{12}}$:

- Actor=T. Hanks→output=Drama (θ=0.75)

Obviously, output "Drama" is predicted as the output for input x_{12}. Therefore, $\mathcal{F}$=$\{Drama\}$. In the next iteration, another rule is extracted from $\mathcal{S}^{x_{12}}$:

- Actor=T. Hanks$\wedge$output=Drama$\rightarrow$output=Crime (θ=0.50)

Since no more rules can be extracted from $\mathcal{S}^{x_{12}}$, the process stops. Outputs "Drama" and "Crime" are predicted for input x_{12}. In summary, outputs "Romance" and "Crime" are equaly related to feature "Actor=T. Hanks". Therefore, it may be difficult to distinguish these two outputs based solely on this feature. However, if we are confident that a movie starred by "T. Hanks" should be classified as "Drama", then it is more likely that this movie should be classified as "Crime", rather than "Romance".

5.1.3 Empirical Results

In this section we will present the experimental results for the evaluation of the proposed multi-label classification algorithms, namely: LAC-MR-IO and LAC-MR-CO.

Setup – In all the experiments we used 10-fold cross-validation and the final results of each experiment represent the average of the ten runs. All the results to be presented were found statistically significant based on a t-test at 95% confidence level.

Computational Environment – The experiments were performed on a Linux-based PC with a Intel Pentium III 1.0 GHz processor and 1 GB RAM.

Evaluation Criteria – The evaluation of multi-label classification algorithms is much more complicated than the evaluation of single-label ones. We used three evaluation criteria that were proposed in [Schapire and Singer, 2000]:

- Hamming Loss (h_{x_i}): Evaluates how many times input x_i is misclassified (i.e., an output not related to x_i is predicted or an output related to x_i is not predicted), as show in Equation 5.1, where p is the number of possible outputs and Δ stands for the symmetric difference between the set of predicted outputs ($\mathcal{P}_{x_i}$) and the set of true outputs ($\mathcal{A}_{x_i}$) for input x_i.

$$h_{x_i} = \frac{1}{p} \mid \mathcal{P}_{x_i} \Delta \mathcal{A}_{x_i} \mid . \tag{5.1}$$

- Ranking Loss (r_{x_i}): Evaluates the average fraction of output pairs (c_j, c_k, for which $c_k \in \mathcal{A}_{x_i}$ and $c_j \notin \mathcal{A}_{x_i}$) that are reversely ordered (i.e., $j > k$), as shown

in Equation 5.2 (where $\overline{\mathcal{A}_{x_i}}$ denotes the complementary set of $\mathcal{A}_{x_i}$).

$$r_{x_i} = \frac{\mid \{(c_k, c_j) \in \mathcal{A}_{x_i} \times \overline{\mathcal{A}_{x_i}} : j > k\} \mid}{\mid \mathcal{A}_{x_i} \mid\mid \overline{\mathcal{A}_{x_i}} \mid}. \tag{5.2}$$

- One-Error (o_{x_i}): Evaluates the label ranking performance from a restrictive perspective as it only determines if the top-ranked label is present in the set of proper outputs ($\mathcal{A}_{x_i}$) of input x_i, as shown in Equation 5.3.

$$o_{x_i} = \begin{cases} 0 & \text{if most likely output is in } \mathcal{A}_{x_i} \\ 1 & \text{otherwise.} \end{cases} \tag{5.3}$$

The overall classification performance is obtained by averaging each criterion, that is:

$$\text{Hamming Loss} = \frac{1}{\mid \mathcal{T} \mid} \times \sum_{x_i \in \mathcal{T}} h_{x_i} \tag{5.4}$$

$$\text{Ranking Loss} = \frac{1}{\mid \mathcal{T} \mid} \times \sum_{x_i \in \mathcal{T}} r_{x_i} \tag{5.5}$$

$$\text{One Error} = \frac{1}{\mid \mathcal{T} \mid} \times \sum_{x_i \in \mathcal{T}} o_{x_i} \tag{5.6}$$

The ACM Digital Library

Two collections were used in the experiments. The first collection, which is called ACM-DL (first level), was extracted from the first level of the ACM Computing Classification System (`http://portal.acm.org/dl.cfm/`), comprising a set of 81,251 documents labeled using the 11 first level categories of ACM. The second collection, ACM-DL (second level) contains the same set of documents of ACM-DL (first level), but these documents are labeled using the 81 second level categories. In both collections, each document is described by its title and abstract, citations, and authorship, resulting in a huge and sparse feature space. For ACM-DL (first level), the average number of labels (or outputs) for each document is 2.55, while for ACM-DL (second level) the average number of labels for each document is 2.82.

Baselines – The evaluation is based on a comparison involving ML-SVM [Elisseeff and Weston, 2001].

Parameters – For ML-SVM, polynomial kernels of degree 8 were used. For LAC-MR-IO and LAC-MR-CO, σ_{min} was set to 0.01. For LAC-MR-IO, Δ_{min} was set to 0.25.

Analysis – Table 5.2 shows categorization performance for different classification algorithms. Best results, including statistical ties, are shown in bold. ML-SVM and LAC-MR-IO shown competitive performance, and LAC-MR-CO is the best performer. To verify if the association between labels was properly explored by LAC-MR-CO, we checked wether the explicitly correlated categories shown in the ACM Computing Classification System (`http://www.acm.org/class/1998/overview.html`) were indeed used. We verified that some of these explicitly correlated categories often appear together in the predicted label combination (i.e., *Files* and *Database Management*, or *Simulation/Modeling* and *Probability/Statistics*). We further verified that some of the associated labels appear more frequently in the predictions performed by LAC-MR-CO than it was observed in the predictions performed by the other algorithms.

	First Level			Second Level		
Algorithms	Hamming Loss	Ranking Loss	One-Error	Hamming Loss	Ranking Loss	One-Error
ML-SVM	0.225	0.194	**0.244**	0.327	0.299	0.348
LAC-MR-IO	0.222	0.216	**0.238**	0.319	0.294	**0.331**
LAC-MR-CO	**0.187**	**0.179**	**0.238**	**0.285**	**0.273**	**0.331**

Table 5.2. Categorization performance for different algorithms.

Gene Functional Analysis

Genes play a fundamental role in life. Thus, predicting the function of a certain gene is of great interest. In this section we evaluate LAC-MR-IO and LAC-MR-CO for sake of predicting the gene functional classes of the *Yeast Saccharomyces cerevisiae*, which is one of the best studied organisms. More specifically, the YEAST dataset [Schapire and Singer, 2000] is investigated. The whole set of functional classes is structured into hierarchies up to 4 levels deep 4. In our evaluation, only functional classes in the top hierarchy are considered. The dataset is composed of a set of 2,417 genes. Each gene is described by the concatenation of micro-array expression data and phylogenetic profile, and is associated with a set of functional classes. There are 14 possible class labels (functions), and the average number of labels for each gene is 4.24.

Baselines – The evaluation is based on a comparison involving BoosTexter [Schapire and Singer, 2000], ADTBoost.MH [Comité et al., 2003], and ML-SVM [Elisseeff and Weston, 2001]. We believe that these methods are representative of some of the most effective multi-label methods available.

Parameters – For BoosTexter and ADTBoost.MH, the number of boosting rounds was set to 500 and 50, respectively. For ML-SVM, polynomial kernels of degree 10 were used. For LAC-MR-IO and LAC-MR-CO, σ_{min} was set to 0.01. For LAC-MR-IO, Δ_{min} was set to 0.25.

Analysis – Table 5.3 shows the results. Best results, including statistical ties, are shown in bold. The YEAST dataset is considered complex, with strong dependencies among labels. LAC-MR-CO provide gains of 24% in terms of one-error, considering BoosTexter as the baseline. The reason is that the simple decision function used by BoosTexter is not suitable for this complex dataset. Also, LAC-MR-IO and LAC-MR-CO are able to explore many more associations than ADTBoost.MH. LAC-MR-CO performs much better than ML-SVM since it is able to explore dependencies between labels.

Algorithms	Hamming Loss	Ranking Loss	One-Error
BoosTexter	0.220	0.186	0.278
ADTBoost.MH	0.207	–	0.244
ML-SVM	0.196	0.163	0.217
LAC-MR-IO	0.191	0.164	**0.213**
LAC-MR-CO	**0.179**	**0.150**	**0.213**

Table 5.3. Categorization performance for different algorithms.

5.1.4 Summary

In this section we introduced multi-label demand-driven associative classification algorithms, LAC-MR-IO and LAC-MR-CO. The functions produced by these algorithms map inputs to multiple outputs. LAC-MR-IO is a very simple extension of LAC-MR, which produces functions that predict all outputs that are associated with the input (a threshold, Δ_{min}, defines the necessary degree of association). LAC-MR-CO exploits the association between different outputs to refine the mapping function. Our experiments, concerning classification problems such as document categorization and gene functional analysis, have shown that these algorithms are able to provide gains that are up to 24%.

5.2 Multi-Metric Classification

The classification performance of an associative classification algorithm is strongly dependent on the statistic measure or metric that is used to quantify the strength of the association between features and classes (i.e., confidence, correlation etc.). Previous studies have shown that classification algorithms produced using different metrics may predict conflicting outputs for the same input, and that the best metric to use is data-dependent and rarely known while designing the algorithm [Veloso et al., 2009c]. This uncertainty concerning the optimal match between metrics and problems is a dilemma, and prevents associative classification algorithms to achieve their maximal performance.

A possible solution to this dilemma is to exploit the competence, expertise, or assertiveness of classification algorithms produced using different metrics. The basic idea is that each of these algorithms has a specific sub-domain for which it is most competent (i.e., there is a set of inputs for which this algorithm consistently provides more accurate predictions than algorithms produced using other metrics). Particularly, we investigate stacking-based meta-learning methods, which use the training data to find the domain of competence of associative classification algorithms produced using different metrics. The result is a set of competing algorithms that are produced using different metrics. The ability to detect which of these algorithms is the most competent one for a given input leads to new algorithms, which are denoted as *competence-conscious associative classification algorithms*.

5.2.1 Related Work

A variety of related methods, that combine several algorithms, has already been proposed. Well known methods include bagging [Breiman, 1996], boosting [Schapire, 1999], and stacking [Wolpert, 1992]. In the following, we will focus our attention on stacking methods, since the algorithms to be proposed in this section are most related to them. Stacking is based on the idea that different classification algorithms provide different but complementary explanations of the training data. Thus, the predictions of these different (base) algorithms provide novel information that can be used as meta-features to build a new training data. Then, a meta-classifier is produced using this new training data, but instead of predicting the correct output for a given input, the meta-classifier predicts the base algorithm that is most likely to correctly predict the output for such input. The obvious advantage, in this case, is that the errors of a base algorithm may be counter-attacked by the hits of other constituent algorithms.

The integration of algorithms using methods related to stacking was largely ex-

plored [Ferri et al., 2004; Ortega et al., 2001; Tsymbal et al., 2006; Gama and Brazdil, 2000; Antonie et al., 2006]. We believe that the work of Ortega et. al [Ortega et al., 2001] is the closest to ours. They used a referee (which in our case is a meta-classifier) to indicate the best constituent algorithm to be applied for each input. The approach used to produce the referee (which is based on decision trees) is different to the approach we used to produce the meta-classifier.

Self-delegation [Ferri et al., 2004] is another method for combining the predictions of different base algorithms, and thus it is also related to this work. The idea is that each base algorithm chooses by itself for which inputs it can safely predicts the output. This choice is based on the confidence in its prediction. A base algorithm delegates the difficult or uncertain predictions to other algorithms. Clearly, this strategy produces algorithms which are exclusively defined in terms of the original features (no meta-features are generated). This simplicity may be desirable, but it may neglect important information associated with meta-features.

Several statistic metrics can be used to estimate the association between inputs and outputs [Tan et al., 2002; Lavrac et al., 1999; Hilderman and Hamilton, 2001], but the most competent one is rarely known in advance. Thus, we propose to explore the diversity among classification algorithms that are produced using different metrics to boost the classification performance of the final algorithm [Veloso et al., 2009c] (which will be refereed as a competence-conscious associative algorithm).

5.2.2 Algorithms

Next we present three algorithms for multi-metric associative classification. The first one, which will be referred to as LAC-MR-SD (standing for LAC-MR with self-delegation), delegates the metric to be used for producing $f_S^{x_i}$. The second one, which will be referred to as LAC-MR-OC (standing for LAC-MR with output-centric metric selection), groups the competence of metrics according to the outputs. The last one, which will be referred to as LAC-MR-IC (standing for LAC-MR with input-centric metric selection), is much finer-grained and associates the competence of metrics to inputs.

Metrics

Next, we present several metrics for measuring the strength of association between a set of features ($\mathcal{X}$) and classes ($c_1, c_2, \ldots, c_p$). Some of these metrics are popular ones [Agrawal et al., 1993; Tan et al., 2002], while others were recently used in the context of associative classification[Arunasalam and Chawla, 2006]. These metrics interpret

association using different definitions. We believe that these definitions are different enough to indicate that the corresponding algorithms may present some diversity.

- Confidence (γ_1) [Agrawal et al., 1993]: This metric was defined in Equation 3.1. Its value ranges from 0 to 1.

- Added Value (γ_2) [Hilderman and Hamilton, 2001]: This metric measures the gain in accuracy obtained by using rule $\mathcal{X} \rightarrow c_j$ instead of always predicting c_j, as shown in Equation 5.7. Negative values indicate that always predicting c_j is better than using the rule. Its value ranges from -1 to 1.

$$\gamma_2 = p(c_j|\mathcal{X}) - p(c_j) \tag{5.7}$$

- Certainty (γ_3) [Lavrac et al., 1999]: This metric measures the increase in accuracy between rule $\mathcal{X} \rightarrow c_j$ and always predicting c_j, as shown in Equation 5.8. It assumes values smaller than 1.

$$\gamma_3 = \frac{p(c_j|\mathcal{X}) - p(c_j)}{p(\overline{c_j})} \tag{5.8}$$

- Yules'Q (γ_4) and Yules'Y (γ_5) [Tan et al., 2002]: These metrics are based on odds value, as shown in Equations 5.9 and 5.10, respectively. Their values range from -1 to 1. The value 1 implies perfect positive association between $\mathcal{X}$ and c_j, value 0 implies no association, and value -1 implies perfect negative association.

$$\gamma_4 = \frac{p(\mathcal{X} \cup c_j) \times p(\overline{\mathcal{X} \cup c_j}) - p(\mathcal{X} \cup \overline{c_j}) \times p(\overline{\mathcal{X}} \cup c_j)}{p(\mathcal{X} \cup c_j) \times p(\overline{\mathcal{X} \cup c_j}) + p(\mathcal{X} \cup \overline{c_j}) \times p(\overline{\mathcal{X}} \cup c_j)} \tag{5.9}$$

$$\gamma_5 = \frac{\sqrt{p(\mathcal{X} \cup c_j) \times p(\overline{\mathcal{X} \cup c_j})} - \sqrt{p(\mathcal{X} \cup \overline{c_j}) \times p(\overline{\mathcal{X}} \cup c_j)}}{\sqrt{p(\mathcal{X} \cup c_j) \times p(\overline{\mathcal{X} \cup c_j})} + \sqrt{p(\mathcal{X} \cup \overline{c_j}) \times p(\overline{\mathcal{X}} \cup c_j)}} \tag{5.10}$$

- Strength Score (γ_6) [Arunasalam and Chawla, 2006]: This metric measures the correlation between $\mathcal{X}$ and c_j, but it also takes into account how $\mathcal{X}$ is correlated to the complement of c_j (i.e., $\overline{c_j}$), as shown in Equation 5.11. Its value ranges from 0 to ∞.

$$\gamma_6 = \frac{p(\mathcal{X}|c_j) \times p(c_j|\mathcal{X})}{p(\mathcal{X}|\overline{c_j})} \tag{5.11}$$

- Support (γ_7) [Agrawal et al., 1993]: This metric was defined in Equation 3.2. Its value ranges from 0 to 1.

- Weighted Relative Confidence (γ_8) [Lavrac et al., 1999]: This metric trades off accuracy and generality, as shown in Equation 5.12. The first component is the accuracy gain that is obtained by using rule $\mathcal{X} \rightarrow c_j$ instead of always predicting c_j. The second component incorporates generality.

$$\gamma_8 = (p(c_j|\mathcal{X}) - p(c_j)) \times p(\mathcal{X}) \tag{5.12}$$

Although we focus our analysis only on these metrics, the algorithms to be introduced are general and able to exploit any number of metrics, transparently.

LAC-MR-SD – Selecting an appropriate metric is a major issue while designing an associative classification algorithm. Algorithms produced by different metrics often present different classification performance. Depending on the characteristics of the problem, some metrics may be more suitable than others. Given a set composed of algorithms, $\mathcal{C}_{\gamma_1}$, $\mathcal{C}_{\gamma_2}$, ..., $\mathcal{C}_{\gamma_q}$, which were produced using different metrics, we must select which algorithm is the one most likely to perform a correct prediction. Equation 3.5 can be used to estimate the reliability of a prediction, and this information can be used to select the most reliable prediction performed, considering all constituent classification algorithms. This is the approach used by LAC-MR-SD, which is illustrated in Algorithm 10. For a given input x_i, the predicted output is the one which is associated with the highest likelihood $\hat{p}(c_j|x_i)$ amongst all competing algorithms. The basic idea is to use the most reliable prediction (among the predictions performed by all competing algorithms) to select the output for x_i.

Although simple, LAC-MR-SD does not exploit the competence of each constituent algorithm. In fact, each base algorithm simply decides by itself the inputs for which it will predict the output, not meaning that the selected inputs belong to its domain of competence.

Domain of Competence

The optimal match between metrics and problems is valuable information. In this section we present an approach to estimate such matching. The proposed approach may be viewed as an application of Wolpert's stacked generalization [Wolpert, 1992]. From a general point of view, stacking can be considered a meta-learning method, as it refers to the induction of algorithms over inputs that are, in turn, the predictions of other algorithms induced from the training data.

Algorithm 10 Finding $f_{\mathcal{S}}^{x_i}$, according to LAC-MR-SD.

Require: The training data $\mathcal{S}$, and an input $x_i \in \mathcal{T}$
Ensure: $f_{\mathcal{S}}^{x_i}$

$\mathcal{S}^{x_i} \Leftarrow \mathcal{S}$ projected according to x_i
$\mathcal{R}^{x_i} \Leftarrow$ rules $\mathcal{X} \rightarrow c_j$ extracted from $\mathcal{S}^{x_i}$
for each competing algorithm $\mathcal{C}_{\gamma_q}$ **do**
  produce candidate functions $c_{\gamma_q \mathcal{S}}^{x_i}$ using rules in $\mathcal{R}^{x_i}$
end for
return $f_{\mathcal{S}}^{x_i}$ which is the function that provides the highest likelihood $\hat{p}(c_j|x_i)$, amongst all candidate functions $c_{\gamma_q \mathcal{S}}^{x_i}$

Algorithm 11 Enhancing the training data with the competence of each competing algorithm.

Require: The original training data $\mathcal{S}$, and a cross-validation parameter k
Ensure: The enhanced training data $\mathcal{S}_e$

split $\mathcal{S}$ into k partitions, so that $\mathcal{S}=\{d_1 \cup d_2 \cup \ldots \cup d_k\}$
$\mathcal{S}_e \Leftarrow \emptyset$
for each partition d_t **do**
  for each input $x_i \in d_t$ **do**
    $\gamma \Leftarrow \emptyset$
    $\{\mathcal{S} - d_t\}^{x_i} \Leftarrow \{\mathcal{S} - d_t\}$ projected according to x_i
    $\mathcal{R}^{x_i} \Leftarrow$ rules $\mathcal{X} \rightarrow c_j$ extracted from $\{\mathcal{S}\text{-}d\}^{x_i}$
    build different algorithms, $\mathcal{C}_{\gamma 1}^t, \mathcal{C}_{\gamma 2}^t, \ldots, \mathcal{C}_{\gamma q}^t$, using rules in $\mathcal{R}^{x_i}$
    for each algorithm $\mathcal{C}_{\gamma_j}$ **do**
      if $\mathcal{C}_{\gamma_j}$ correctly predicts the class for x_i **then**
        $\gamma \Leftarrow \gamma \cup \gamma_j$
      end if
    end for
    $\mathcal{S}_e \Leftarrow \mathcal{S}_e \cup \{(x_i, y_i) \cup \gamma\}$
  end for
end for

The process starts by enhancing the original training data using the outputs predicted by the base algorithms, $\mathcal{C}_{\gamma_1}$, $\mathcal{C}_{\gamma_2} \ldots \mathcal{C}_{\gamma_q}$. Algorithm 11 shows the basic steps involved in the process. Initially, the enhanced training data, $\mathcal{S}_e$ is empty. An example x_i, along with the competence of each algorithm with regard to x_i (i.e., which competing algorithm correctly predicted the output for x_i), is inserted into $\mathcal{S}_e$. The process continues until all examples are processed. In the end, for each example $x_i \in \mathcal{S}_e$ we have a list of competing algorithms that predicted the correct output for x_i, and this information enables learning the domains of competence of each algorithm.

Example – To illustrate the process, consider the example shown in Tables 5.4 and 5.5. Table 5.4 shows the original training data, $\mathcal{S}$. Using the process described in Algorithm 11, the competence of each algorithm with regard to each input is appended to $\mathcal{S}$, resulting in the enhanced training data, $\mathcal{S}_e$, which is shown in Table 5.5. In this case, for a given example x_i, metric γ_j is shown if the corresponding algorithm $\mathcal{C}_{\gamma_j}$ has correctly predicted the output for input x_i using the stacking procedure. The enhanced training data, $\mathcal{S}_e$, can be exploited in several ways. In particular, we will use $\mathcal{S}_e$ to produce competence-conscious associative classification algorithms, as it will be discussed next.

id	output	input a_1 a_2 ... a_l
1	c_1	1 3 ... 6
2	c_1	1 3 ... 7
3	c_1	2 4 ... 6
4	c_2	2 4 ... 7
5	c_2	2 5 ... 8
6	c_2	2 4 ... 6
7	c_3	1 3 ... 9
8	c_3	2 5 ... 9
9	c_3	2 4 ... 8
10	c_3	2 4 ... 9

Table 5.4. Training data given as an example of multi-metric problem.

id	output	input a_1 a_2 ... a_l	Competent Metric(s) (per instance)	Most Competent Metric(s) (per class)
1	c_1	1 3 ... 6	γ_2	γ_1
2	c_1	1 3 ... 7	γ_1 γ_3	
3	c_1	2 4 ... 6	γ_1	
4	c_2	2 4 ... 7	γ_1 γ_2	γ_1
5	c_2	2 5 ... 8	γ_1 γ_2 γ_3	
6	c_2	2 4 ... 6	γ_1	
7	c_3	1 3 ... 9	γ_2	γ_2
8	c_3	2 5 ... 9	γ_2 γ_3	
9	c_3	2 4 ... 8	γ_1 γ_2 γ_3	
10	c_3	2 4 ... 9	γ_2	

Table 5.5. Enhanced training data, $\mathcal{S}_e$.

Competence-Conscious Metric Selection

In this section we present algorithms that exploit $\mathcal{S}_e$ to produce functions $f_{\mathcal{S}}^{x_i}$. The challenge, in this case, is to properly select a competent metric for a specific input. The competence-conscious algorithms to be presented differ in how they perform the analysis of the domains of competence of the competing algorithms.

LAC-MR-OC – The competence of algorithms produced using different metrics are often associated with certain outputs (or classes). Some metrics, for instance, produce algorithms which show preference for more frequent classes, while others produce algorithms which show preference for less frequent ones. As an illustrative example, please consider Table 5.5. Algorithm derived from metric γ_1 is extremely competent for inputs that are related to outputs c_1 and c_2. On the other hand, if we consider inputs that are related to c_3, the algorithm derived from metric γ_2 perfectly classifies all inputs. This information (which is shown in the last column of Table 5.5) may be used to produce output-centric competence-conscious algorithms. The process is depicted in Algorithm 12. It starts with a meta-classifier, $\mathcal{M}$, which learns the most competent base algorithm for a given class. Specifically, instead of extracting rules $\mathcal{X} \rightarrow c_j$, the meta-classifier extracts rules $\mathcal{X} \rightarrow \gamma_i$, which maps features (i.e., in the third column of Table 5.5) to metrics (i.e., in the fifth column of Table 5.5). Then, for each input $x_i \in \mathcal{T}$, the meta-classifier indicates the most competent base algorithm, $\mathcal{C}_{\gamma_j}$, that is then used to produce $f_{\mathcal{S}}^{x_i}$.

Algorithm 12 Finding $f_{\mathcal{S}}^{x_i}$, according to LAC-MR-OC.

Require: The enhanced training data $\mathcal{S}_e$ (i.e., the 3^{rd} and 5^{th} columns of Table 5.5), and an input $x_i \in \mathcal{T}$
Ensure: $f_{\mathcal{S}}^{x_i}$

1: $\mathcal{S}_e^{x_i} \Leftarrow \mathcal{S}_e$ projected according to x_i
2: **for** each metric γ_t **do**
3: $\quad \mathcal{R}_{\gamma_t}^{x_i} \Leftarrow$ rules $\mathcal{X} \rightarrow \gamma_t$ extracted from $\mathcal{S}_e^{x_i}$
4: $\quad$ estimate $\hat{p}(\gamma_t|x_i)$, according to Equation 3.5
5: **end for**
6: let $\mathcal{C}_{\gamma_j}$, such that $\hat{p}(\gamma_j|x_i) \geq \hat{p}(\gamma_t|x_i) \forall t \neq j$, be the most competent algorithm for input x_i
7: **r**eturn $f_{\mathcal{S}}^{x_i}$ which is produced by $\mathcal{C}_{\gamma_j}$

LAC-MR-IC – Although the competence of some base algorithms are associated with certain classes, specific inputs may be better classified using other base algorithms. In such cases, a finer-grained analysis of competence is desired. As an illustrative

example, consider again Table 5.5. Although algorithm derived from metric γ_1 is the most competent one to predict the outputs for inputs that are related to class c_1, algorithm derived from metric γ_2 is the only one which competently classifies input 1 (which is related to c_1). Again, a meta-classifier, $\mathcal{M}$, is used to explore such cases. The process is depicted in Algorithm 13. In this case, the meta-classifier learns the most competent metric by extracting rules of the form $\mathcal{X} \rightarrow \gamma_j$, which maps features (i.e., in the third column of Table 5.5) to metrics (i.e., in the fourth column of Table 5.5). Then, for each input $x_i \in \mathcal{T}$, the meta-classifier indicates the most competent base algorithm, $\mathcal{C}_{\gamma_j}$, that is then used to produce $f_{\mathcal{S}}^{x_i}$.

Algorithm 13 Finding $f_{\mathcal{S}}^{x_i}$, according to LAC-MR-IC.

Require: The enhanced training data $\mathcal{S}_e$ (i.e., the 3^{rd} and 4^{th} columns of Table 5.5), and an input $x_i \in \mathcal{T}$

Ensure: $f_{\mathcal{S}}^{x_i}$

1: $\mathcal{S}_e^{x_i} \Leftarrow \mathcal{S}_e$ projected according to x_i
2: **for e**ach metric γ_t **do**
3: $\quad \mathcal{R}_{\gamma_t}^{x_i} \Leftarrow$ rules $\mathcal{X} \rightarrow \gamma_t$ extracted from $\mathcal{S}_e^{x_i}$
4: $\quad$ estimate $\hat{p}(\gamma_t|x_i)$, according to Equation 3.5
5: **end for**
6: let $\mathcal{C}_{\gamma_j}$ such that $\hat{p}(\gamma_j|x_i) \geq \hat{p}(\gamma_t|x_i) \forall t \neq j$, be the most competent algorithm for input x_i
7: **r**eturn $f_{\mathcal{S}}^{x_i}$ which is produced by $\mathcal{C}_{\gamma_j}$

The main advantage of LAC-MR-OC and LAC-MR-IC is that, in practice, multiple metrics produce competent algorithms for a particular input x_i, but $\mathcal{M}$ needs to predict only one of them (competent algorithms are not mutually exclusive, and thus, in practice, multiple metrics produce competent algorithms for x_i). This redundancy in competence that exists when different metrics are taken into account, may increase the chance of selecting a competent algorithm.

5.2.3 Empirical Results

In this section we will present the empirical results for the evaluation of the proposed multi-metric classification algorithms, which include LAC-MR-SD, LAC-MR-OC, and LAC-MR-IC.

Setup – In all the experiments we used 10-fold cross-validation and the final results of each experiment represent the average of the ten runs. All the results to be presented were found statistically significant based on a t-test at 95% confidence level.

Baselines – The evaluation is based on a comparison involving SVM algorithms and against ER (standing for External Referee), which is a combination method proposed in [Ortega et al., 2001] (in this case, the competing algorithms are $\mathcal{C}_{\gamma_1}, \ldots, \mathcal{C}_{\gamma_8}$, but the most competent algorithm for each input is selected using a decision tree referee). For SVM, we used the implementation available at `http://svmlight.joachims.org/` (version 3.0). We used our own implementation of ER.

Computational Environment – The experiments were performed on a Linux-based PC with a Intel Pentium III 1.8 GHz processor and 1 GB RAM.

Bounds for multi-metric associative classification – We derived simple lower and upper bounds for the classification performance of LAC-MR-SD, LAC-MR-OC, and LAC-MR-IC. The lower bound is the performance that is obtained by randomly selecting a competent algorithm. Clearly, this lower bound increases with the redundancy between the base algorithms (this redundancy exists because competent algorithms are not mutually exclusive, and, thus, for a particular input x_i, multiple base algorithms may be competent). The upper bound is the classification performance that would be obtained by an oracle which always predicts a competent base algorithm (note that perfect performance is not always possible, since it may not exist a competent algorithm for some inputs). Clearly, this upper bound increases with the accuracy and diversity associated with base algorithms.

The ACM Digital Library

In this section we will evaluate the proposed algorithms using a collection of documents extracted from the ACM digital library. This is the same collection used in the experiments shown in Section 3.3.2. There 6,682 documents, which were labeled under 8 first level categories of ACM, namely: Hardware (C1), Computer Systems Organization (C2), Software (C3), Computing Methodologies (C4), Mathematics of Computing (C5), Information Systems (C6), Theory of Computation (C7), Computing Milieux (C8).

Parameters – As suggested by the grid parameter search tool in LibSVM [Chang and Lin, 2001], polynomial kernel of degree 6 was used in the experiments.

Evaluation Criteria – Categorization performance for the various methods being evaluated, is expressed through MicF_1.

Analysis – Using the rules extracted from ACM-DL, we can analyze the relationship between the widely used confidence metric (γ_1) with other metrics, as shown in Figure 5.1 (to ease the observation of this relationship, we also include, in each graph, a thicker line which indicates the corresponding confidence value). Each point in the graphs corresponds to a rule, for which it is shown the values of some metrics (i.e., confidence in the x-axis and another metric in the y-axis). Different lines are associated with different outputs (i.e., classes). Clearly, each metric has its particular behavior with varying values of confidence. We will use these relationships to understand some of the results to be presented. For lower values of confidence, Added Value (γ_2) has a preference for less frequent classes, but, after a certain confidence value, the preference is for more frequent classes. Certainty (γ_3) always prefer less frequent classes, but linearly approaches confidence as its value increases. Yules'Q (γ_4) and Yules'Y (γ_5) have a similar behavior, showing preference for less frequent classes and hardly penalizing associations with low confidence values. Strength Score (γ_6) and Weighted Relative Confidence (γ_8) both prefer less frequent classes, but Strength Score shows a non-proportional preference for associations with higher values of confidence. The relationship between confidence and support (γ_7) is omitted, but, by definition, support shows a preference for more frequent classes.

Table 5.6 shows the classification performance obtained by different base algorithms. Best results, including statistical ties, are shown in bold. We will first analyze the performance associated with each category, and then the final classification performance, which is shown in the last line of the table. Algorithms produced by confidence ($\mathcal{C}_{\gamma_1}$) and support ($\mathcal{C}_{\gamma_7}$) performed very well in the most frequent categories (Software, Information Systems and Theory of Computer Science). On the other hand, inputs belonging to less frequent categories (Computer Methodologies, Mathematics of Computer Science, and Computer Science Organization) were better classified using algorithms produced by Yules'Q ($\mathcal{C}_{\gamma_4}$) and Yules'Y ($\mathcal{C}_{\gamma_5}$). This is expected, and is in agreement with the behaviors depicted in Figure 5.1 (algorithms produced by Yules'Y and Yules'Q show a preference for less frequent categories). The best base algorithm is the one that better balances its performance over all categories. Although the $\mathcal{C}_{\gamma_5}$ algorithm was not the best one for any specific category of ACM-DL, it was the best overall base algorithm.

Table 5.7 shows classification performance for multi-metric algorithms. Best results, including statistical ties, are shown in bold. LAC-MR-SD shows a performance that is similar to the performance obtained by most of the base algorithms (the improvement, when it exists, is only marginal). Competence-conscious algorithms LAC-MR-OC and LAC-MR-IC showed the best performances. LAC-MR-IC outperformed all other pro-

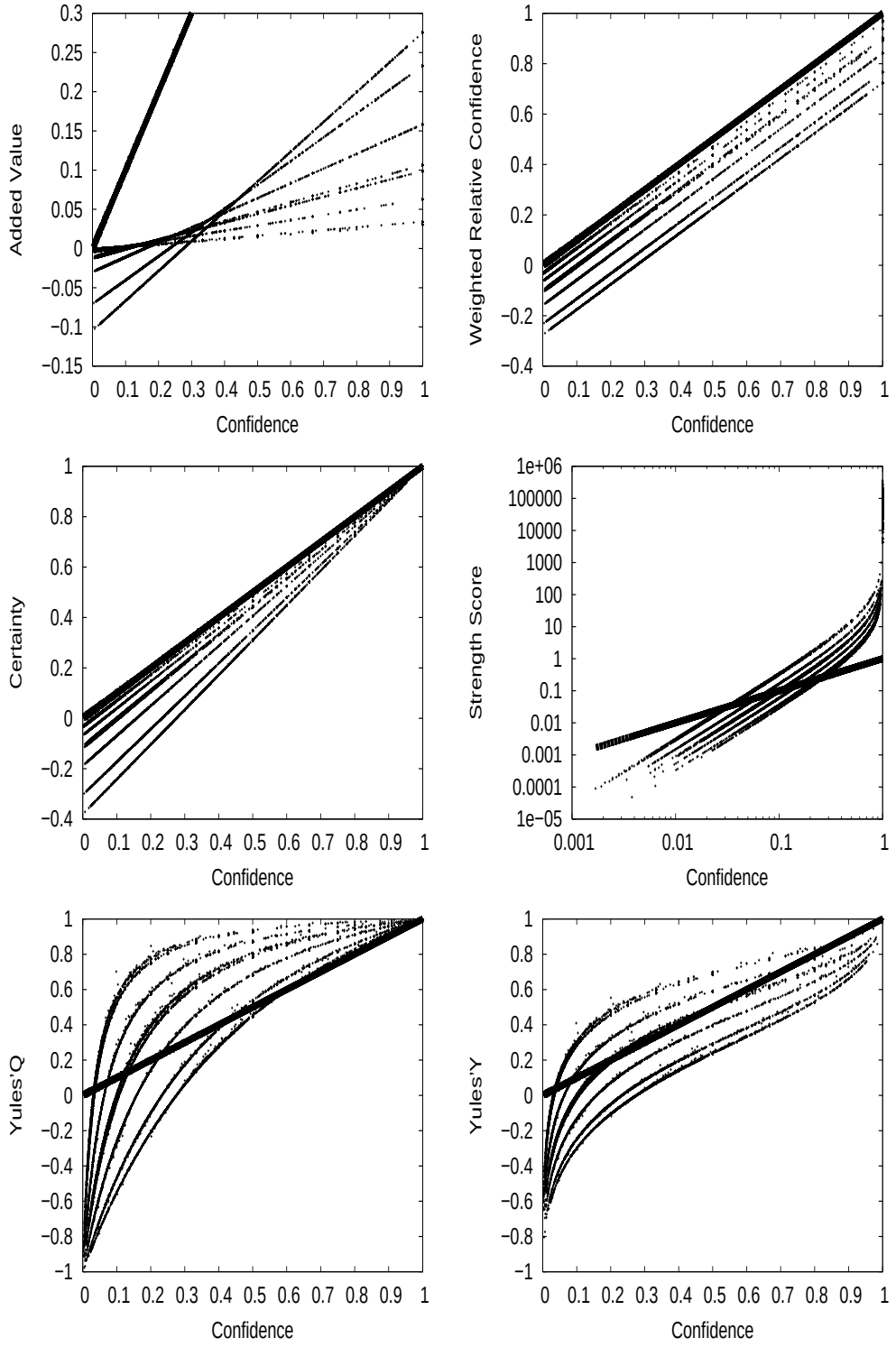

Figure 5.1. Relationship between confidence and other metrics.

Category	$\mathcal{C}_{\gamma_1}$	$\mathcal{C}_{\gamma_2}$	$\mathcal{C}_{\gamma_3}$	$\mathcal{C}_{\gamma_4}$	$\mathcal{C}_{\gamma_5}$	$\mathcal{C}_{\gamma_6}$	$\mathcal{C}_{\gamma_7}$	$\mathcal{C}_{\gamma_8}$
C1	0.809	**0.846**	0.826	0.834	0.834	**0.848**	0.183	0.628
C2	0.714	0.785	0.758	0.772	**0.799**	0.752	0.313	0.785
C3	0.912	0.851	0.888	0.871	0.864	0.748	**0.960**	0.880
C4	0.569	**0.690**	0.628	0.657	0.661	0.676	0.090	0.547
C5	0.548	0.624	0.593	**0.675**	**0.680**	**0.670**	0.010	0.329
C6	**0.948**	0.929	**0.937**	**0.931**	0.927	0.893	0.689	0.761
C7	**0.922**	0.893	0.897	0.890	0.887	0.889	0.507	0.687
C8	0.641	0.715	0.687	0.721	0.729	**0.755**	0.071	0.481
Total	0.843	0.847	**0.850**	**0.852**	**0.855**	0.810	0.566	0.735

Table 5.6. Classification performance of base algorithms.

posed algorithms and baselines, providing gains of more than 7%, when compared against SVM, and gains of more than 8.5% when compared against ER. LAC-MR-IC is always far superior than the corresponding lower bound, but it is also relatively far from the corresponding upper bound.

	Lower	LAC-MR			Upper		
Category	Bound	SD	OC	IC	Bound	ER	SVM
C1	0.715	0.813	0.809	**0.821**	0.893	0.801	0.729
C2	0.723	0.730	0.738	0.766	0.880	0.719	**0.879**
C3	0.870	0.876	0.884	**0.918**	0.983	0.874	0.661
C4	0.562	0.581	**0.623**	**0.623**	0.795	0.604	0.515
C5	0.563	0.568	0.625	0.648	0.751	0.613	**0.907**
C6	0.877	**0.919**	0.911	**0.925**	0.965	0.898	0.869
C7	0.837	**0.906**	0.895	**0.902**	0.922	0.876	0.672
C8	0.591	0.654	0.697	0.697	0.823	0.674	**0.771**
Total	0.798	0.848	0.858	**0.881**	0.925	0.811	0.827

Table 5.7. Classification performance of multi-metric algorithms.

We also performed an analysis on how the different base algorithms were used by LAC-MR-OC and LAC-MR-IC, as can be seen in Figure 5.2. LAC-MR-OC utilized only few base algorithms, specially $\mathcal{C}_{\gamma_2}$, $\mathcal{C}_{\gamma_3}$, and $\mathcal{C}_{\gamma_7}$. Metric γ_4 was used to produce algorithms to only one category, and metrics γ_5 and γ_8 were not used to produce any base algorithm at all (this is because the corresponding algorithms were not the most competent in any category of ACM, and therefore are not considered by LAC-MR-OC). LAC-MR-IC, on the other hand, utilized all base algorithms, specially $\mathcal{C}_{\gamma_1}$, $\mathcal{C}_{\gamma_2}$ and $\mathcal{C}_{\gamma_3}$. Both LAC-MR-OC and LAC-MR-IC make large utilization of base algorithms $\mathcal{C}_{\gamma_2}$ and $\mathcal{C}_{\gamma_3}$. For LAC-MR-OC, some areas of expertise can be easily detected. Base algorithm $\mathcal{C}_{\gamma_2}$ is considered competent for categories Hardware and Computer Science

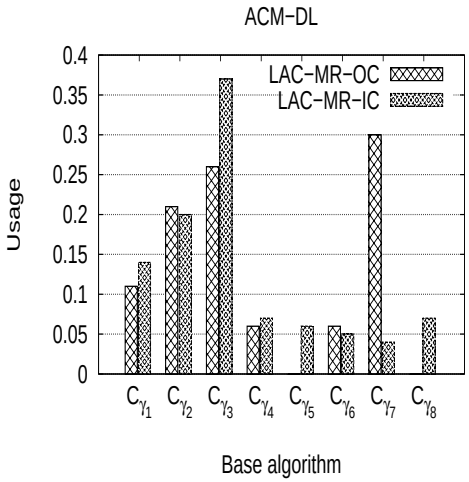

Figure 5.2. Utilization of base algorithms.

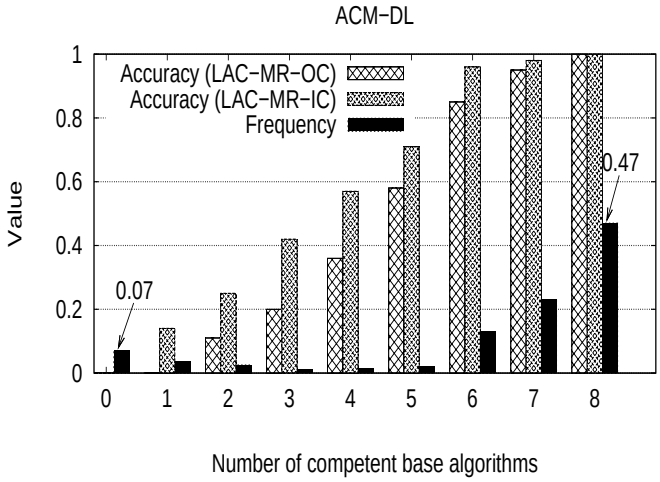

Figure 5.3. Distribution of competent algorithms.

Organization, while $\mathcal{C}_{\gamma_3}$ is considered competent for category Information Systems. For LAC-MR-IC, areas of expertise are finer grained, but with manual inspection we detected that $\mathcal{C}_{\gamma_1}$ is considered competent for category Computer Science Organization, and $\mathcal{C}_{\gamma_3}$ is considered competent for category Milieux.

We finalize this set of experiments by analyzing one of the reasons of the good performance showed by LAC-MR-IC. Figure 5.3 shows the accuracy associated with scenarios for which a different number of base algorithms are competent. The frequency of occurrence of each scenario is also shown (note that both accuracy and frequency values are shown in the y-axis). As it can be seen, for more than 7% of the inputs in the test set, no base algorithm is competent, and, obviously, these inputs were misclassified (this means that the inclusion of other metrics may improve classification

performance in this dataset). As expected, accuracy increases with the number of competent base algorithms. For almost half of the inputs in the test set all base algorithms were competent. In these scenarios, there is no risk of misclassification, since a base algorithm produced by any metric will perform a correct prediction. The accuracy associated with scenarios where only 7 and only 6 of the base algorithms are competent, is also extremely high (respectively, 99% and 96%). These three scenarios (i.e., 8, only 7, and only 6 base algorithms are simultaneously competent) correspond to 86% of the inputs, and the average accuracy associated with these three scenarios is almost 98% for LAC-MR-IC. Further, LAC-MR-IC shows to be more robust than LAC-MR-OC, providing superior accuracy (relative to the accuracy of LAC-MR-OC) in scenarios where only few base algorithms are competent.

Web Spam Detection

In this application the objective is to detect malicious actions aimed at the ranking functions used by search engines. We used a dataset obtained from the Web Spam Challenge (`http://webspam.lip6.fr/wiki/pmwiki.php`). The dataset is very skewed (only 6% of the examples are spam pages). Each example is composed of direct features (i.e., number of pages in the host, number of characters in the host name etc.) link-based features (i.e., in-degree, out-degree, PageRank etc.) and content-based features (i.e., number of words in the page, average word length etc.).

Parameters – As suggested by the grid parameter search tool in LibSVM [Chang and Lin, 2001], we used a linear kernel with parameter C set to 5.00.

Evaluation Criteria – For this application, classification performance is computed through MicF_1 and MacF_1 measures, and the area under the ROC curve [Fürnkranz and Flach, 2003].

Analysis – Table 5.8 shows the classification performance obtained by different base algorithms. Best results, including statistical ties, are shown in bold. $\mathcal{C}_{\gamma_1}$ and $\mathcal{C}_{\gamma_7}$ showed impressive performance in terms of MicF_1. This is expected, because the vast majority of examples are legitimate pages, and confidence and support have preference for more frequent classes. On the other hand, $\mathcal{C}_{\gamma_1}$ and $\mathcal{C}_{\gamma_7}$ showed poor classification performance in terms of MacF_1 and AUC (i.e., no spam pages were detected). The remaining base algorithms were able to detect some spam pages, specially $\mathcal{C}_{\gamma_6}$, which also shows impressive performance in terms of accuracy. In terms of AUC, $\mathcal{C}_{\gamma_2}$ and $\mathcal{C}_{\gamma_3}$ showed the best classification performance, amongst all base algorithms. Thus,

different algorithms produced by different metrics show distinct performance depending on the evaluation target (i.e., $MicF_1$, $MacF_1$, or AUC).

Evaluation Target	$\mathcal{C}_{\gamma_1}$	$\mathcal{C}_{\gamma_2}$	$\mathcal{C}_{\gamma_3}$	$\mathcal{C}_{\gamma_4}$	$\mathcal{C}_{\gamma_5}$	$\mathcal{C}_{\gamma_6}$	$\mathcal{C}_{\gamma_7}$	$\mathcal{C}_{\gamma_8}$
$MicF_1$	**0.946**	0.704	0.702	0.894	0.901	**0.948**	**0.946**	0.880
$MacF_1$	0.486	0.522	0.522	0.584	**0.589**	**0.592**	0.486	**0.587**
AUC	0.500	**0.756**	**0.756**	0.607	0.606	0.562	0.500	0.629

Table 5.8. Classification performance of base algorithms.

Table 5.9 shows classification performance for multi-metric algorithms. Best results, including statistical ties, are shown in bold. LAC-MR-OC and LAC-MR-IC were the best performers in terms of $MacF_1$. Although LAC-MR-IC showed to be far from the optimal classification performance, it showed impressive gains when compared against SVM and ER, in terms of $MacF_1$ and AUC.

Evaluation Target	Lower Bound	LAC-MR SD	LAC-MR OC	LAC-MR IC	Upper Bound	ER	SVM
$MicF_1$	0.852	0.861	0.870	0.897	0.990	0.866	**0.956**
$MacF_1$	0.588	0.594	0.609	**0.624**	0.947	0.586	0.504
AUC	0.662	0.730	0.718	**0.789**	0.908	0.725	0.512

Table 5.9. Classification performance of multi-metric algorithms.

5.2.4 Summary

In this section we introduced multi-metric demand-driven associative classification algorithms, LAC-MR-OC and LAC-MR-IC. These algorithms combine predictions performed by mapping functions that are produced using different association metrics. A specific mapping function is selected to predict the output for each input in the test set. The selection is based on the domain of competence of each algorithm, which is the set of inputs that are accurately classified by the algorithm. The proposed multi-metric algorithms differ on the granularity of the domain of competence. LAC-MR-IC employs a finer-grained analysis of competence, and, as shown in our experiments concerning classification problems such as document categorization and Web spam detection, it is able to provide gains of more than 8.5% in classification performance.

5.3 Calibrated Classification

Given an input x_i and an arbitrary output c_j, a classification algorithm usually works by estimating the probability of x_i being related to c_j (i.e., class membership probability). Well calibrated classification algorithms are those able to produce functions that provide accurate estimates of class membership probabilities, that is, the estimated probability $\hat{p}(c_j|x_i)$ is close to $p(c_j|\hat{p}(c_j|x_i))$, which is the true, (unknown) empirical probability of x_i being related to output c_j given that the probability estimated by the classification algorithm is $\hat{p}(c_j|x_i)$. Calibration is not a necessary property for producing an accurate approximation of the target function, and, thus, most of the research has focused on direct accuracy maximization strategies rather than on calibration. However, non-calibrated functions are problematic in applications where the reliability associated with a prediction must be taken into account (i.e., cost-sensitive classification, cautious classification etc.). In these applications, a sensible use of the classification algorithm must be based on the reliability of its predictions, and thus, the algorithm must produce well calibrated functions.

5.3.1 Related Work

There are studies investigating calibration of classification algorithms, such as SVMs, Naive Bayes, and Decision Trees. The calibration of Naive Bayes and Decision Tree classification algorithms were investigated in [Zadrozny and Elkan, 2001], where it was shown that probabilities estimated by these algorithms are usually far from the observed, true probabilities. The same methodology was used to show that SVM algorithms are poorly calibrated [Zadrozny and Elkan, 2002], and that the distortion very often forms a sigmoid pattern. Boosting-based algorithms were shown to produce functions that are poorly calibrated in [Niculescu-Mizil and Caruana, 2005]. Although these algorithms do not produce calibrated functions, they tend to assign higher probabilities to the correct output. Therefore, predictions are usually accurate. The advantages of calibrated functions were discussed in [Cohen and Goldszmidt, 2004], where it was shown that calibrating a function is guaranteed not to decrease its classification performance.

There are several existing methods used to correct the distortion between estimated and observed probabilities. Methods for calibrating SVM algorithms transform the predictions to posterior probabilities by passing them through a sigmoid. The parametric approach proposed in [Platt, 1999] consists in finding the parameters a and b for a sigmoid function of the form $\hat{p}_c(c_i|x) = {}^{1}/_{(1+\exp\{a\hat{p}(c_i|x)+b\})}$, which transforms the original estimated probability, $\hat{p}(c_i|x)$, into a calibrated estimate, $\hat{p}_c(c_i|x)$. These parameters

are found by minimizing the negative log-likelihood of the training data. Methods for calibrating functions produced by Decision Tree and Naive Bayes algorithms were proposed in [Zadrozny and Elkan, 2001]. These methods are based on smoothing the distribution of the original estimates, and they also rely on finding parameters from the training data. Functions produced by boosting based algorithms are calibrated using a method called Logistic Regression, proposed in [Friedman et al., 2000]. This method transforms original estimates, $\hat{p}(c_i|x)$, into calibrated estimates, $\hat{p}_c(c_i|x)$, using the function $\hat{p}_c(c_i|x) = {}^{1}/_{(1+\exp\{-2a\hat{p}(c_i|x)\})}$. All the mentioned methods find the corresponding parameters through 10-fold cross-validation using the training data.

5.3.2 Algorithms

In this section we define calibrated algorithms, and then we propose methods to calibrate associative classification algorithms. Further, we present two algorithms that are calibrated using the proposed calibration methods. The first one, which will be referred to as LAC-MR-NC (standing for LAC-MR with naive calibration), is calibrated using a naive calibration method. The second one, which will be referred to as LAC-MR-EM (standing for LAC-MR with entropy minimization), is calibrated using a sophisticated calibration method based on entropy minimization.

τ-calibrated Classification Algorithms

The calibration of a classification algorithm can be visualized using reliability diagrams. Diagrams for two arbitrary algorithms using an arbitrary dataset are depicted in Figure 5.4 (Left). These diagrams are built as follows [DeGroot and Fienberg, 1982]. First, the probability space (i.e., the x-axis) is divided into a number of bins, which was chosen to be 10 in our case. Probability estimates (i.e., theoretical probabilities) with value between 0 and 0.1 fall in the first bin, estimates with value between 0.1 and 0.2 fall in the second bin, and so on. The fraction of correct predictions associated with each bin, *which is the true, empirical probability* (i.e., $p(c|\hat{p}(c_j|x_i))$), is plotted against the estimated probability (i.e., $\hat{p}(c_j|x_i)$). If the algorithm is well calibrated, the points will fall near the diagonal line, indicating that estimated probabilities are close to empirical probabilities. The degree of calibration of an algorithm, denoted as τ, is obtained by measuring the discrepancy between observed probabilities (o_i) and estimated probabilities (e_i), as shown in Equation 5.13 (where k is the number of bins). Values of τ range from 0 to 1. A value of 0 means that there is no relationship between estimated probabilities and true probabilities. A value of 1 means that all points lie exactly on a straight line with no scatter. Algorithm 1 is better calibrated than

Algorithm 2, as shown in Figure 5.4 (Right).

$$\tau = 1 - \frac{1}{k}\sum_{i=1}^{k}\frac{(o_i - e_i)^2}{(o_i + e_i)^2} \tag{5.13}$$

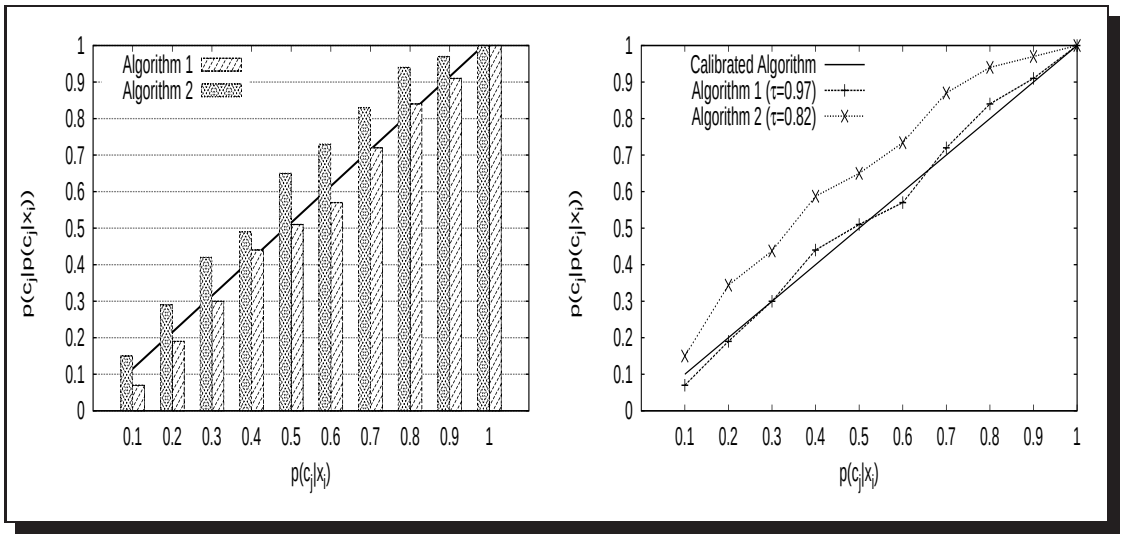

Figure 5.4. Reliability diagram and τ-calibrated algorithms.

LAC-MR-NC – To transform original probability estimates, $\hat{p}(c_j|x_i)$, into accurate well calibrated probabilities, we also use a method based on binning. The method starts by estimating membership probabilities using the training data, $\mathcal{S}$. A typical method is 10-Fold Cross-Validation. In this case, $\mathcal{S}$ is divided into 10 partitions, and at each trial, 9 partitions are used for training, while the remaining partition is used to simulate a test set. After the 10 trials, the algorithm will have stored in the set $\mathcal{O}$, the membership probability estimates for all inputs in $\mathcal{S}$. This process is shown in Algorithm 14.

Once the probabilities are estimated, a naive calibration method would proceed by first sorting these probabilities in ascending order (i.e., the probability space), and then dividing them into k equal-sized bins, each having pre-specified boundaries. An estimate is placed in a bin according to its value (i.e., values between 0 and $\frac{1}{k}$ are placed in the first bin, values between $\frac{1}{k}$ and $\frac{2}{k}$ in the second, and so on). The probability associated with a bin is given by the fraction of correct predictions that were placed in it. An estimate $\hat{p}(c_j|x_i)$ is finally calibrated by using the probability associated with the corresponding bin. Specifically, each bin $b_{l\hookrightarrow u} \in \mathcal{B}$ (with l and u being its boundaries) works as a map, relating estimates $\hat{p}(c_j|x_i)$ (such that $l \leq \hat{p}(c_j|x_i) < u$) to the corresponding calibrated estimates, $p_{b_{l\hookrightarrow u}}$. Thus, this process essentially discretizes

Algorithm 14 Estimating membership probabilities.

Require: Examples in $\mathcal{S}$

Ensure: For each input x_i in $\mathcal{S}$, the corresponding membership probabilities $\hat{p}(c_1|x_i), \hat{p}(c_2|x_i), \ldots, \hat{p}(c_p|x_i)$, along with the correct output

1: $\mathcal{O} \Leftarrow \emptyset$
2: Split $\mathcal{S}$ into 10 equal-sized partitions, $p_1, p_2, \ldots, p_{10}$
3: **for** each partition p_j **do**
4: **for** each input $x_i \in p_j$ **do**
5: Estimate probabilities, $\hat{p}(c_1|x_i), \hat{p}(c_2|x_i), \ldots, \hat{p}(c_p|x_i)$, using $\{\mathcal{S}\text{-}p_i\}$ as training
6: $\mathcal{O} \Leftarrow \mathcal{O} \cup \{(\hat{p}(c_1|x_i), v_1)\} \cup \ldots \cup \{(\hat{p}(c_p|x_i), v_p)\}$, where v_j=1 if c_j is the correct class for example x_i, and v_j=0 otherwise
7: **end for**
8: **end for**
9: **r**eturn $\mathcal{O}$

the probability space into intervals, so that the accuracy associated with the predictions in each interval is as reliable as possible.

Such naive method, however, may be disastrous as critical information may be lost due to innapropriate bin boundaries. Instead, we propose to use information entropy associated with candidate bins to select the boundaries [Fayyad and Irani, 1993].

LAC-MR-EM – This algorithm uses the information in $\mathcal{O}$ to initially find a threshold that minimizes the entropy over all possible partitions; and it is then recursively applied to both of the partitions induced by the threshold. To illustrate the method, suppose we are given a set of pairs $(\hat{p}(c_j|x_i), v)$[1] $\in \mathcal{O}$. In this case, the entropy of $\mathcal{O}$ is given by Equation 5.14.

$$\begin{aligned} E(\mathcal{O}) = &-\frac{|(\hat{p}(c_j|x_i), 0) \in \mathcal{O}|}{|\mathcal{O}|} \times \log\left(\frac{|(\hat{p}(c_j|x_i), 0) \in \mathcal{O}|}{|\mathcal{O}|}\right) \\ &-\frac{|(\hat{p}(c_j|x_i), 1) \in \mathcal{O}|}{|\mathcal{O}|} \times \log\left(\frac{|(\hat{p}(c_j|x_i), 1) \in \mathcal{O}|}{|\mathcal{O}|}\right) \end{aligned} \tag{5.14}$$

There is a threshold f, which is a boundary that induces two partitions of $\mathcal{O}$ ($b_{f\leq}$ and $b_{f>}$, where $b_{f\leq}$ contains pairs $(\hat{p}(c_j|x_i), v)$ for which $\hat{p}(c_j|x_i) \leq f$, and $b_{f>}$ contains pairs for which $\hat{p}(c_j|x_i) > f$). The selected threshold, t, is the one which minimizes the weighted average entropies, given by Equation 5.15.

$$E(\mathcal{O}, f) = \frac{|b_{f\leq}|}{|\mathcal{O}|} \times E(b_{f\leq}) + \frac{|b_{f>}|}{|\mathcal{O}|} \times E(b_{f>}) \tag{5.15}$$

[1] v can take the values 0 (the prediction is wrong) or 1 (otherwise), as shown in step 6 of Algorithm 14.

This method is then applied recursively to both of the partitions induced by t, $b_{t\leq}$ and $b_{t>}$, creating multiple intervals until a stopping criterion is fulfilled. Splitting stops if the information gain (the difference between the entropies before and after the split) is lower than the minimum description length [Rissanen, 1978] of the partition, and the final set of bins, $\mathcal{B}$, is found. According to Fayyad and Irani [1993], the minimum description length induced by a threshold t over a partition $\mathcal{O}$ is the second term of the following inequality:

$$E(\mathcal{O}) - E(\mathcal{O}, t) > \frac{\log(|\mathcal{O}| - 1)}{|\mathcal{O}|} + \frac{\Delta(\mathcal{O}, t)}{|\mathcal{O}|} \tag{5.16}$$

where $\Delta(\mathcal{O}, t) = \log(3^k - 2) - (k \times E(\mathcal{O}) - k_1 \times E(b_{t\leq}) - k_2 \times E(b_{t>}))$, and k_i is either 1 or 2 ($k_i = 1$ if the corresponding partition is pure, that is, if it contains only correct (or only incorrect) predictions; and $k_i = 2$ otherwise).

Finally, for each input x_i in the test set, $\hat{p}(c_1|x_i), \hat{p}(c_2|x_i), \ldots, \hat{p}(c_p|x_i)$ are estimated using $\mathcal{S}$. Then, the estimated probabilities are calibrated using the accuracy associated with the appropriate bin in $\mathcal{B}$, as shown in Algorithm 15.

Algorithm 15 Calibrating the probabilities.

Require: Examples in $\mathcal{S}$, inputs in $\mathcal{T}$, the calibrated probability $p_{b_{l\hookrightarrow u}}$ of each bin $b_{l\hookrightarrow u}$
Ensure: For each estimate $\hat{p}(c_j|x_i)$, the corresponding calibrated estimate $\hat{p}_c(c_j|x_i)$

1: **for e**ach input $x_i \in \mathcal{T}$ **do**
2: Estimate probabilities, $\hat{p}(c_1|x_i), \hat{p}(c_2|x_i), \ldots, \hat{p}(c_p|x_i)$, using $\mathcal{S}$ as training
3: **f**or each c_j **d**o output $\hat{p}_c(c_j|x_i)$=$p_{b_{l\hookrightarrow u}}$, such that $l \leq \hat{p}(c_j|x_i) < u$
4: **end for**

	Category	Features
d_1	Databases	Rules in Database Systems
d_2	Databases	Applications of Logic Databases
d_3	Databases	Hypertext Databases and Data Mining
d_4	Data Mining	Mining Association Rules in Large Databases
d_5	Data Mining	Database Mining: A Performance Perspective
d_6	Data Mining	Algorithms for Mining Association Rules
d_7	Inf. Retrieval	Text Databases and Information Retrieval
d_8	Inf. Retrieval	Information Filtering and Information Retrieval
d_9	Inf. Retrieval	Term Weighting Approaches in Text Retrieval
d_{10}	Inf. Retrieval	Performance of Information Retrieval Systems

Table 5.10. Example using documents of a digital library.

Example – Table 5.10 shows an illustrative example composed of 10 documents extracted from a digital library. Each document belongs to one category. Such documents are given as training data. Several rules are extracted from these documents. Specifically, $\mathcal{R}^{d_1}$, which is the set of rules (extracted from $\{\mathcal{S}-p_1\}$) matching document d_1, includes:

1. text=system(s)→Inf. Retrieval (θ=1.00)
2. text={rule(s)∧database(s)}→Data Mining (θ=1.00)
3. text=rule(s)→Data Mining (θ=1.00)
4. text=database(s)→Databases (θ=0.40)
5. text=database(s)→Data Mining (θ=0.40)
6. text=database(s)→Inf. Retrieval (θ=0.20)

From such decision rules, class membership probabilities for document d_1 are estimated using Equation 3.5, resulting in the following probabilities: $\hat{p}$(Databases|d_1)=0.19, $\hat{p}$(Data Mining|d_1)=0.44, and $\hat{p}$(Inf. Retrieval|d_1)=0.37. Membership probabilities for all documents are shown in Table 5.11, where the number between parenthesis indicates if the prediction is correct (1) or not (0).

	$\hat{p}$(Databases|d)	$\hat{p}$(Data Mining|d)	$\hat{p}$(Inf. Retrieval|d)
d_1	0.19 (1)	0.44 (0)	0.37 (0)
d_2	0.27 (1)	0.48 (0)	0.24 (0)
d_3	0.26 (1)	0.61 (0)	0.13 (0)
d_4	0.36 (0)	0.46 (1)	0.18 (0)
d_5	0.27 (0)	0.25 (1)	0.48 (0)
d_6	0.30 (0)	0.53 (1)	0.17 (0)
d_7	0.22 (0)	0.25 (0)	0.53 (1)
d_8	0.00 (0)	0.29 (0)	0.71 (1)
d_9	0.00 (0)	0.00 (0)	1.00 (1)
d_{10}	0.31 (0)	0.31 (0)	0.38 (1)

Table 5.11. Class membership probabilities.

Figure 5.5 (Left) shows the process of setting bin boundaries by entropy minimization, for category "Data Mining". Initially, the probability space (which ranges from 0.00 to 1.00) is divided into two bins. The cut point at 0.45 gives an information gain which is higher than the minimum description length of initial bin. Now, there are two bins. The bin on the right (i.e., [0.45-1.00]) is not divided anymore, since additional

cut points would not provide enough information gain. The bin on the left is further divided into two other bins. The cut point at 0.20 gives an information gain which is higher than the minimum description length of this bin. Then, the process stops because no more bins are created. Figure 5.5 (Right) shows the same process of setting bin boundaries for category "Inf. Retrieval".

Bins obtained for each category are shown in Table 5.12. For this simplified example, only two bins are produced for category "Databases", and only three bins are produced for categories "Data Mining" and "Inf. Retrieval". The calibrated probability for each bin, which is the fraction of correct predictions within each bin, are also shown in Table 5.12.

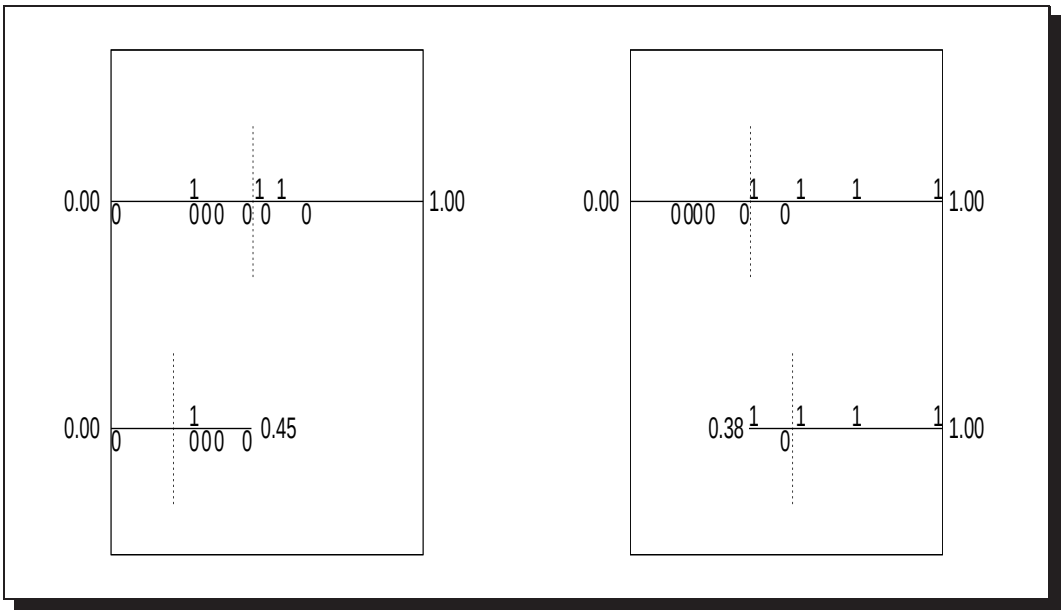

Figure 5.5. Calculating bin boundaries for different categories (category "Data Mining" on the left, and category "Inf. Retrieval" on the right).

Databases		Data Mining		Inf. Retrieval	
Boundaries	Prob.	Boundaries	Prob.	Boundaries	Prob.
[0.00-0.17]	0.000	[0.00-0.20]	0.000	[0.00-0.38]	0.000
[0.17-1.00]	0.375	[0.20-0.45]	0.200	[0.38-0.52]	0.500
		[0.45-1.00]	0.500	[0.52-1.00]	1.000

Table 5.12. Bin boundaries and calibrated probabilities for each category.

5.3.3 Empirical Results

In this section we present the experimental results for the evaluation of the proposed calibrated algorithms, LAC-MR-NC (which is calibrated using the naive calibration method) and LAC-MR-EM (which is calibrated using the MDL-based entropy minimization method).

Computational Environment – The experiments were performed on a Linux-based PC with a Intel Pentium III 1.0 GHz processor and 1 GB RAM.

Baselines – The evaluation is based on a comparison against current state-of-the-art calibrated algorithms, which include SVM [Boser et al., 1992], Naive-Bayes [Cussens, 1993], and Decision Tree classifiers [Quinlan, 1993]. After being calibrated using specific methods [Platt, 1999; Cestnik, 1990; Zadrozny and Elkan, 2001], these algorithms are respectively referred to as CaSVM, CaNB, and CaDT.

The ACM Digital Library

For this application we used the ACM-DL dataset, which was described in Section 3.3.2. The classification algorithm must decide to which category a document belongs. However, the administrator of the digital library imposes an additional minimum accuracy requirement, acc_{min}, to the algorithm. In this case, the algorithm must estimate the total accuracy after each prediction is performed, and then it must decide to continue classifying documents (if the estimated accuracy is higher than acc_{min}) or to stop classification (if the estimated accuracy is lower than acc_{min}).

Setup – In all the experiments we used 10-fold cross-validation and the final results of each experiment represent the average of the ten runs. All the results to be presented were found statistically significant based on a t-test at 95% confidence level.

Evaluation Criteria – We used accuracy, τ, and the fraction of documents classified at acc_{min}, to assess classification performance.

Parameters – For CaSVM we used linear kernels and set C=0.90. These parameters were set according to the grid parameter search tool in LibSVM [Chang and Lin, 2001]. For CaNB and CaDT we used the default parameters, which were also used in other works [Cussens, 1993]. For LAC-MR-NC, the number of bins was set to 5. For LAC-MR-NC and LAC-MR-EM, we set σ_{min}=0.001.

Analysis − The bins produced by different calibration methods are shown in Figure 5.6. Bin boundaries are shown in the x-axis and the corresponding calibrated probabilities are shown in the y-axis. Coincidentally, the MDL-based Entropy-Minimization method also produced 5 bins, but with varying sizes.

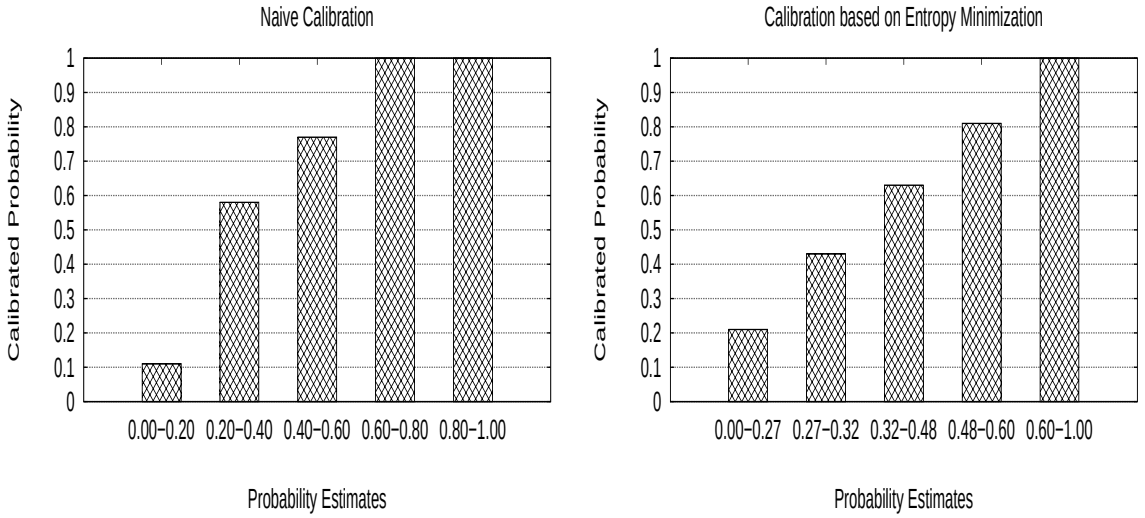

Figure 5.6. Bins produced for category "Information Systems".

After the bins were found, we apply LAC-MR to the test set, and we replace the original probability estimate (x-axis) by the calibrated probability associated with the corresponding bin (y-axis). The result of calibration is depicted in Figure 5.7, which shows τ values for LAC-MR, before and after being calibrated with different methods (resulting in LAC-MR-NC and LAC-MR-EM algorithms). Other algorithms were also evaluated. The worst algorithm in terms of calibration is SVM with τ=0.69. After calibrating SVM, the corresponding algorithm, CaSVM, shows τ=0.75. NB and LAC-MR, with τ=0.76 and τ=0.78, respectively, are already better calibrated than CaSVM. These algorithms, when calibrated, show the best calibration degrees − CaNB with a τ=0.91, and LAC-MR-EM with τ=0.97. Next, we will evaluate how this difference in calibration affects the effectiveness of the algorithms.

We continue our analysis by evaluating each algorithm in terms of its ability for estimating the actual accuracy. Figure 5.8 shows the actual accuracy and the accuracy estimates obtained with each algorithm, so that the corresponding values can be directly compared[2]. As expected, LAC-MR-EM shows to be better calibrated than LAC-MR-NC. This is because the bins used by LAC-MR-NC are produced in an ad-hoc way, while the bins used by LAC-MR-EM are produced following the entropy-minimization strategy. The direct consequence of applying such method is that a bin is likely to con-

[2]For each experiment, predictions were sorted from the most reliable to the least reliable.

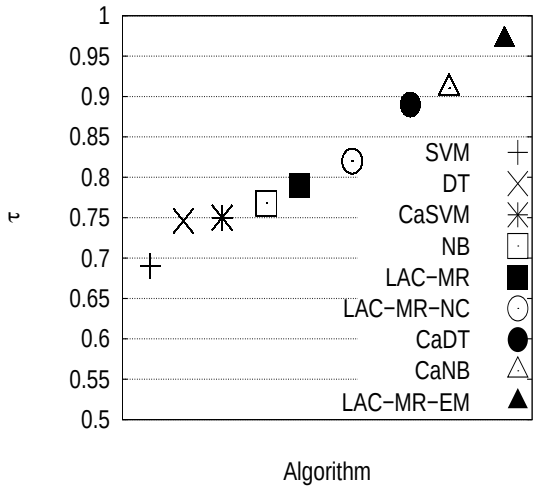

Figure 5.7. Algorithms, before and after being calibrated.

tain predictions which are as similar as possible. While in most of the cases CaNB and CaDT are well calibrated, CaSVM very often underestimates or overestimates the actual accuracy, and is thus poorly calibrated. The main reason of the poor performance of CaSVM is that Platt Scaling is prone to overfitting, since this calibration method is based on regression. The other calibration mechanisms apparently do not overfit as much. This explanation is supported by the results present in [Cohen and Goldszmidt, 2004] (which show that Naive Bayes algorithms are much better calibrated than SVM algorithms).

If the administrator of the digital library specifies a threshold acc_{min} (i.e., the minimum acceptable accuracy of the algorithm), then the value of the algorithm resides in how many documents it is able to classify while respecting acc_{min}. Figure 5.8 shows the fraction of documents in the test set each algorithm is able to classify for a given value of acc_{min} (y-axis). Clearly, LAC-MR-EM is the best performer, except for acc_{min} values higher than 0.95, when the best performer is LAC-MR-NC. CaNB and CaDT are in close rivalry, with CaDT being slightly superior. In most of the cases, both CaNB and CaDT show to be superior than LAC-MR-NC. CaSVM, as expected, is the worst performer for all values of acc_{min}.

KDDCUP'98

For this application we used a dataset called KDD-98, which was used in KDDCUP'98 contest. This dataset was provided by the Paralyzed Veterans of America (PVA), an organization which devises programs and services for US veterans. With a database of over 13 million donors, PVA is also one of the world's largest direct mail fund raisers.

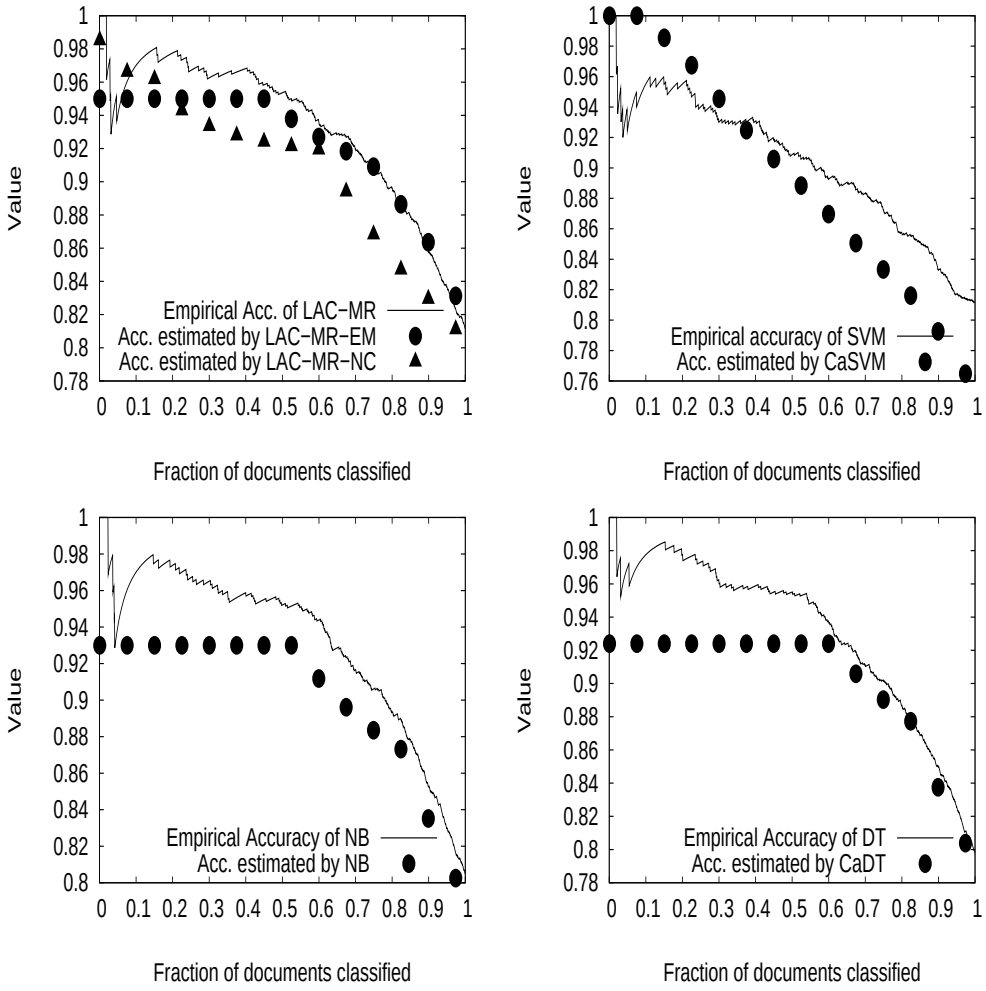

Figure 5.8. Accuracy estimated by calibrated algorithms.

The total cost invested in generating a request (including the mail cost), is \$0.68 per piece mailed. Thus, PVA wants to maximize net revenue by soliciting only individuals that are likely to respond with a donation. The KDD-98 dataset contains information about individuals that have (or have not) made charitable donations in the past. The provided training data consists of 95,412 examples, and the provided test set consists of 96,367 inputs. Each example/input corresponds to an individual, and is composed of 479 features. The training data has an additional field that indicates the amount of the donation (a value \$0 indicates that the individual have not made a donation). From the 96,367 individuals in the test set, only 4,872 are donors. If all individuals in the test set were solicited, the total profit would be only \$10,547. On the other hand, if only those individuals that are donors were solicited, the total profit would be \$72,764. Thus, the classifier must choose which individuals to solicit a new donation from. According to Zadrozny and Elkan [2001], the optimal net maximization strategy is to solicit an individual x_i if and only if $\hat{p}(\text{donate}|x_i) > \frac{0.68}{y(x_i)}$, where $y(x_i)$ is the

expected amount donated by x_i[3]. Thus, in addition to calculating $\hat{p}(\text{donate}|x_i)$, the algorithm must also estimate $y(x_i)$.

Estimating the Donation Amount and $\hat{p}(\textbf{donate}|x)$ – Each donation amount (i.e., \$200, \$199, ..., \$1, and \$0) is considered as an output. Thus, for an individual x_i, rules of the form $\mathcal{X} \rightarrow y$ (with $\mathcal{X} \subseteq x_i$) are extracted from $\mathcal{S}$, and Equation 3.5 is used to estimate the likelihood of each amount (i.e., $\hat{p}(y{=}\$200|x_i), \hat{p}(y{=}\$199|x_i), \ldots, \hat{p}(y{=}\$0|x_i)$). The donation amount, $y(x_i)$, is finally estimated by a linear combination of the probabilities associated with each amount, as shown in Equation 5.17. The probability of donation, $\hat{p}(\text{donate}|x_i)$, is simply given by $1 - \hat{p}(y{=}\$0|x_i)$.

$$y(x_i) = \sum_{i=\$0}^{\$200} i \times \hat{p}(y = i|x_i) \tag{5.17}$$

Parameters

For CaSVM we used linear kernels and set C=2.00. These parameters were set according to the grid parameter search tool in LibSVM [Chang and Lin, 2001]. For CaNB and CaDT we used the default parameters, which were also used in other works [Cussens, 1993]. For LAC-MR-NC, we evaluate four different configurations, with 5, 8, 10, and 15 bins. For LAC-MR-NC and LAC-MR-EM, we set σ_{min}=0.001.

Evaluation Criteria

We used profit as the primary metric for assessing the effectiveness of the algorithms for net revenue optimization. For assessing the accuracy of probability estimates, we use the mean squared error (MSE). Calibration degree, τ, was also used.

Analysis

The calibration degree, τ, achieved by each algorithm, is shown in Figure 5.9. CaSVM, LAC-MR-NC(N15) and LAC-MR-NC(N10) achieved the lowest calibration degrees. This is because the naive calibration method overfitted the training data (i.e., too many bins incur small-sized bins for which the corresponding accuracy may not be reliable, and for CaSVM, the Platt Scaling method is based on regression). LAC-MR-EM and CaNB are the best performers, achieving τ values as high as 0.94. Next we

[3]The basic idea is to solicit a person x_i for whom the expected return $\hat{p}(\text{donate}|x_i)y(x_i)$ is greater than the cost of mailing the solicitation.

will analyze the effectiveness of algorithms with different calibration degrees for net revenue optimization.

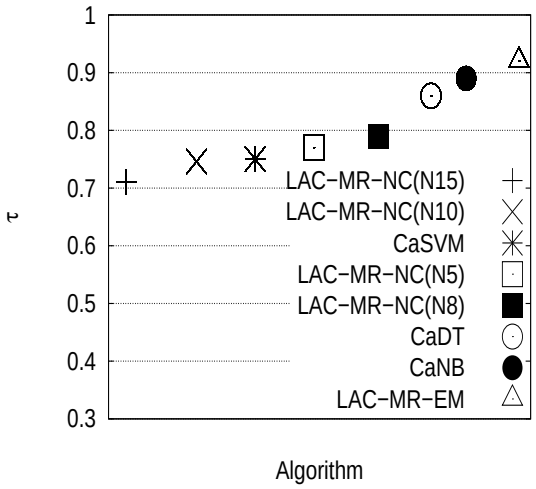

Figure 5.9. Comparing calibration methods in terms of τ.

Algorithm	Profit	MSE
LAC-MR	\$11,097	0.0961
LAC-MR-NC(N8)	\$12,442	0.0958
LAC-MR-EM	\$14,902	0.0934
CaSVM	\$12,969	0.0958
CaNB	\$14,682	0.0952
CaDT	\$14,190	0.0953

Table 5.13. Comparing algorithms in terms of profit and MSE.

Table 5.13 shows the effectiveness of each algorithm. In all cases, the differences in profit are much more accentuated than the differences in MSE. As it can be seen, LAC-MR achieved the lowest profit (which is slightly superior than soliciting all individuals), and this is because it was not calibrated yet. For the same reason, LAC-MR was also the worst performer in terms of MSE. Calibrated algorithms LAC-MR-NC(N8) and CaSVM showed similar performance in terms of profit. According to Cohen and Goldszmidt [2004], the poor performance of CaSVM is, again, due to overfitting. CaDT and CaNB are again in close rivalry. Calibrating LAC-MR using the entropy minimization method is very profitable, and the corresponding algorithm, LAC-MR-EM is the best performer.

5.3.4 Summary

In this section we have introduced calibrated demand-driven associative classification algorithms, LAC-MR-NC and LAC-MR-EM. Calibration is performed using a binning method, which essentialy discretizes the probability space. The first algorithm, LAC-MR-NC, employs a naive strategy in which the number and the boudaries of the bins are explicitly informed by the user. LAC-MR-EM is a more sophisticated algorithm, which employs an entropy-minimization method that recursively splits the bins. Splitting stops when the information gain provided by the split is lower than the minimum description lenght of the bin. As shown in our experiments concerning classification problems such as document categorization and profit optimization, LAC-MR-EM provides gains of more than 17.5%.

5.4 Self-Training

The acquisition of training examples usually requires skilled human annotators to manually label the relationship between inputs and outputs. Due to various reasons, annotators may face inputs that are hard to label. The cost associated with this labeling process thus may render vast amounts of training examples unfeasible. The acquisition of unlabeled inputs (i.e., inputs for which the corresponding output is unknown), on the other hand, is relatively inexpensive. However, it is worthwhile to label at least some inputs, provided that this effort will be then rewarded with an improvement in classification performance. In this section demand-driven associative classification will be extended, so that the corresponding algorithm achieves high classification performance even in the case of limited labeling efforts.

5.4.1 Related Work

Semi-supervised and transductive classification algorithms, are those that incorporate inputs with unkown outputs, into the training data. Semi-supervised algorithms [Blum and Mitchell, 1998; Chapelle et al., 2006] exploit the fact that, frequently, it is unexpensive to collect large amounts of unlabeled inputs (i.e., inputs for which the corresponding outputs are unknown). Transductive algorithms [Vapnik, 1998; Tresp, 2000] explicitly exploit unlabeled inputs in the test set to improve the mapping function.

Semi-supervised and transductive algorithms use few training examples to create more and more pseudo-examples, which are used to produce mapping functions. The basic idea adopted by many of these algorithms is to incorporate predictions

which are likely to be correct (i.e., highly reliable predictions) into the training data [Blum and Chawla, 2001; Bennett and Demiriz, 1998b]. Thus, the training data is progressively enhanced as new unlabeled inputs are processed.

There are several semi-supervised and transductive classification algorithms. Zhu [2008] provides an excellent survey of the main algorithms.

5.4.2 Algorithm

In this section we present a self-training demand-driven associative classification algorithm. This algorithm will be referred to as LAC-MR-ST (standing for LAC-MR with self-training). LAC-MR-ST exploits reliable predictions and the lack of enough evidence supporting the known outputs, to include new examples to $\mathcal{S}$.

Reliable Predictions

A reliable prediction, (x_i, c_j) (where $x_i \in \mathcal{T}$), is the one for which the corresponding class membership probability, $\hat{p}(c_j|x_i)$,is above a given threshold, Δ_{min}. For appropriate values of Δ_{min}, the chance of $y_i \neq c_j$ (i.e., a misclassification) is low, and thus, these predictions may be exploited for the sake of self-training. In this case, a reliable prediction, (x_i, c_j), is considered as a new example and is added to $\mathcal{S}$. Since rules are extracted on a demand-driven basis, the next input to be processed will possibly take advantage of the recently included (pseudo-)example.

Lack of Evidence

Some problems may contain a very large number of outputs. In such cases, it becomes hard for the annotator to specify all the outputs, and the consequence is that some outputs are never explicitly informed in $\mathcal{S}$. The lack of (enough) decision rules predicting any known output present in $\mathcal{S}$, may be exploited to detect the appearance of a novel/unseen output in $\mathcal{T}$. Specifically, for a given input $x_i \in \mathcal{T}$, if the number of rules supporting any known output is smaller than ϕ_{min}, than it is assumed that the input x_i is not related to any output in $\mathcal{S}$. In this case, a new label, c_j, is associated with this possibly new output. The new output, c_j, and the corresponding input, x_i, are considered as a new example, (x_i, c_j), which is included to $\mathcal{S}$.

LAC-MR-ST – This algorithm exploits reliable predictions and the lack of enough evidence to produce novel training examples, which are stored in $\mathcal{N}$.

Naturally, some predictions are not reliable enough for certain values of Δ_{min}. In these cases, such doubtful predictions are abstained. As new examples are included

in $\mathcal{N}$ (i.e., the reliable predictions), they may be exploited, hopefully increasing the reliability of the predictions that were previously abstained. To optimize the usage of reliable predictions, inputs are stored in a priority queue, $\mathcal{Q}$, so that inputs having reliable predictions are considered first. The process works as follows. Initially, inputs in $\mathcal{T}$ are randomly placed in $\mathcal{Q}$. If the output c_j of the input x_i that is located in the beginning of the queue is reliably predicted, then the x_i is removed from $\mathcal{Q}$ and a new example (c_j, x_i) is included into the training data $\mathcal{N}$. Otherwise, if the prediction is not reliable, the corresponding input is simply placed in the end of the queue and it will be processed again only after processing all other inputs. The process continues performing more reliable predictions first, until no more reliable predictions are possible.

The lack of rules supporting any output in $\mathcal{S}$ may be used as evidence indicating the appearance of an output that is not in $\mathcal{S}$. The number of rules that is necessary to consider an output as an already seen one is controlled by a threshold, ϕ_{min}. Specifically, for an input x_i, if the number of rules extracted from $\mathcal{S}^{x_i}$, is smaller than ϕ_{min}, then the output of x_i is considered as an output not in $\mathcal{S}$, and a new label c_j is created to identify such output. Further, this prediction is considered as a new example (x_i, c_j), which is included to $\mathcal{N}$. The basic steps of LAC-MR-ST are shown in Algorithm 16.

5.4.3 Empirical Results

In this section we present experimental results for the evaluation of LAC-MR and LAC-MR-ST.

Setup – In all experiments we used 10-fold cross-validation, and the final results of each experiment represent the average of the five runs. All the results to be presented were found statistically significant based on a t-test at the 95% confidence level.

Evaluation Criteria – Classification performance for the various methods being evaluated is expressed through MicF_1 and MacF_1.

Baselines – We used the k-Way unsupervised Spectral Clustering algorithm as baseline [Han et al., 2005].

Parameters For the K-Way Spectral Clustering algorithm we set k to be the correct number of clusters (thus, the performance reported for this algorithm may be considered as an upper-bound of its true performance). For LAC-MR and LAC-MR-ST we set

Algorithm 16 Including new examples to the original training data.

Require: The training data $\mathcal{S}$, $\mathcal{T}$, σ_{min}, Δ_{min}, and ϕ_{min}
Ensure: $\mathcal{N}$.

1: $\mathcal{Q} \Leftarrow \mathcal{T}$
2: $\mathcal{N} \Leftarrow \mathcal{S}$
3: **for each** input $x_i \in \mathcal{Q}$ **do**
4: $\omega \leftarrow$ false
5: $\mathcal{N}^{x_i} \Leftarrow \mathcal{N}$ projected according to x_i
6: $\mathcal{R}^{x_i} \Leftarrow$ rules $\mathcal{X} \rightarrow c_j$ extracted from $\mathcal{N}^{x_i}$, such that $\pi(\mathcal{X} \rightarrow c_j) \geq \sigma_{min} \times |\mathcal{N}^{x_i}|$
7: **if** $|\mathcal{R}^{x_i}| < \phi_{min}$ **then**
8: create a new label c_k
9: $\mathcal{N} \Leftarrow \mathcal{N} \cup (x_i, c_k)$
10: $\omega \leftarrow$ true
11: **else**
12: **for each** output c_j **do**
13: **if** $\hat{p}(c_j|x_i) \geq \Delta_{min}$ **then**
14: $\mathcal{N} \Leftarrow \mathcal{N} \cup (x_i, c_j)$
15: $\omega \leftarrow$ true
16: **end if**
17: **end for**
18: **end if**
19: **if** $\omega =$false **then**
20: place x_i in the end of the queue, $\mathcal{Q}$
21: **end if**
22: **if** it is not possible to perform reliable predictions anymore **then return** $\mathcal{N}$
23: **end for**

σ_{min}=0.05. Particularly for LAC-MR-ST, we investigated its sensitivity to parameters Δ_{min} and ϕ_{min}.

Computational Environment − The experiments were performed on a Linux-based PC with a Intel Core 2 Duo 1.83 GHz processor and 2 GB RAM.

DBLP and BDBComp

Citations are an essential component of many current digital libraries. Citation management within digital libraries involves a number of tasks. One task in particular, name disambiguation, has required significant attention from the research community due to its inherent difficulty. Name ambiguity in the context of bibliographic citations occurs when one author can be correctly referred to by multiple name variations (synonyms) or when multiple authors have exactly the same name or share the same name variation (polysems). This problem may occur for a number of reasons, including the

lack of standards and common practices, and the decentralized generation of content (e.g., by means of automatic harvesting). Name ambiguity is widespread in many large-scale digital libraries, such as Citeseer, Google Scholar, and DBLP.

Some of the most effective methods seem to be based on the application of supervised machine learning techniques. In this case, the training data consists of examples (x_i, y_i), where x_i is a set of features of a citation, and y_i is a label which identifies the corresponding author. More specifically, such examples are citations for which the correct authorship is known. Although successful cases have been reported [Han et al., 2004], some particular challenges associated with name disambiguation in the context of bibliographic citations, prevent the full potential of supervised machine learning techniques:

- The acquisition of training examples requires skilled human annotators to manually label authors in citations. Annotators may face hard-to-label citations with highly ambiguous authors. The cost associated with this labeling process thus may render vast amounts of examples unfeasible. Thus, classification algorithms must be cost-effective, achieving high classification performance even in the case of limited labeling efforts.

- It is not reasonable to assume that all possible authors are included in the training data (specially due to the scarce availability of examples). Thus, classification algorithms must be able to detect unseen/unknown authors, for whom no label was previously specified.

We used two collections of bibliographic citations. One was extracted from DBLP (`http://dblp.uni-trier.de`) and the other was extracted from BDBComp (`http://www.lbd.ufmg.br/bdbcomp`). Each citation consists of the title of the work, a list of coauthor names, and the title of the publication venue (conference or journal). Pre-processing involved standardizing coauthor names using only the initial letter of the first name along with the full last name, removing punctuation and stop-words of publication and venue titles, stemming publication and venue titles using Porter's algorithm[Porter, 1980], and grouping authors with the same first name initial and the same last name in order to create the ambiguous groups (i.e., groups of citations having different authors with similar names). Table 5.14 shows more detailed information about the collections and their ambiguous groups. Disambiguation is particularly difficult in ambiguous groups such as the C. Chen group, in which the correct author must be selected from 60 possible authors, and in ambiguous groups such as the J. Silva group, in which the majority of authors appears in only one citation.

DBLP			BDBComp		
Ambiguous Group	#Citations	#Authors	Ambiguous Group	#Citations	#Authors
A. Gupta	576	26	A. Oliveira	52	16
A. Kumar	243	14	A. Silva	64	32
C. Chen	798	60	F. Silva	26	20
D. Johnson	368	15	J. Oliveira	48	18
J. Martin	112	16	J. Silva	36	17
J. Robinson	171	12	J. Souza	35	11
J. Smith	921	29	L. Silva	33	18
K. Tanaka	280	10	M. Silva	21	16
M. Brown	153	13	R. Santos	20	16
M. Jones	260	13	R. Silva	28	20
M. Miller	405	12	–	–	–

Table 5.14. The DBLP and BDBComp collections

Analysis In all experiments, we varied the proportion, or fraction of training examples available. For instance, if the fraction of examples available is 0.5, then only half of the examples in the training data was provided to the algorithm. In this case, examples in the training data are randomly selected.

We start our analysis by evaluating the effectiveness of LAC-MR-ST in detecting unseen authors using the BDBComp collection. For each fraction of training examples, we varied ϕ_{min} from 1 to 6. The results are shown in Figure 5.10, where each curve is associated with a different fraction of training examples. For the BDBComp collection, the fraction of unseen authors that are detected increases with ϕ_{min}. This is expected, since the amount of evidence that is required to recognize an author as already seen one, increases for higher values of ϕ_{min}. Further, it becomes more difficult to detect an unseen author when the fraction of training examples increases. This is because, in such cases, (1) more authors are seen (i.e., there are more examples), and (2) there is an increase in the amount of available evidence supporting already seen authors.

We evaluate the effectiveness of LAC-MR-ST in incorporating new training examples using the DBLP collection. For each fraction of training examples, we varied Δ_{min} from 0.5 to 0.9. The results are shown in Figure 5.11. As it can be seen, the performance of LAC-MR-ST decreases when Δ_{min} is set too high (i.e., Δ_{min} >0.75). Further, the performance also decreases when Δ_{min} is set too low (i.e., Δ_{min} <0.65). On one hand, when lower values of Δ_{min} are applied, several citations in the test set, which are associated with wrong predictions, are included in the training data, hurting performance. On the other hand, when higher values of Δ_{min} are applied, only few citations in the test set are included in the training data. For the DBLP collection, LAC-MR-ST

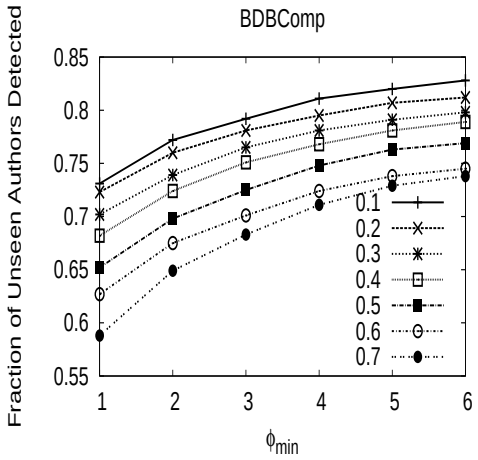

Figure 5.10. Sensitivity to ϕ_{min}.

achieves the best performance when Δ_{min} is between 0.65 and 0.75 (specially when few training examples are available).

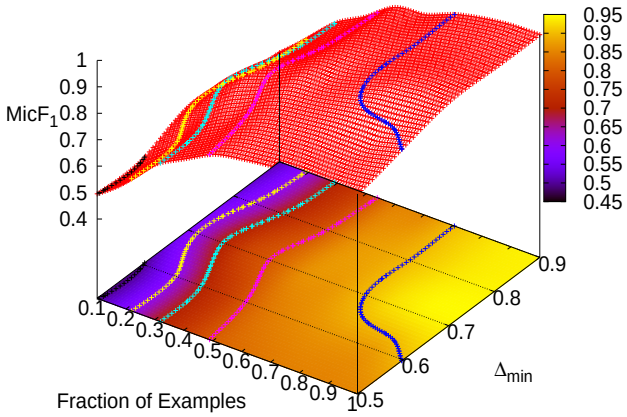

Figure 5.11. Sensitivity to Δ_{min}.

We now evaluate how the self-training ability of LAC-MR-ST improves its performance when compared with LAC-MR. Figure 5.12 shows some of the results. The value associated with each point in each graph is obtained by applying a different combination of Δ_{min} and ϕ_{min}, for different fractions of training examples. For the DBLP collection, gains ranging from 18.4% to 53.8% are observed when few training examples are available. The improvement decreases as more examples are available, since in this

case (1) more authors are seen and (2) additional examples that are included in the training data do not impact so much the final performance. Interestingly, LAC-MR-ST achieves good performance even when not a single example is available for training. This is possible because, in this case, citations authored by unseen authors are included in the training data, and used as training examples. These gains highlight the advantages of self-training.

Improvements obtained using the BDBComp collection are more impressive. This collection contains several authors that appear in only one citation. LAC-MR is not useful in such scenarios. (i.e., if this citation appears in the test set, then the training data contains no evidence supporting the correct author). LAC-MR-ST, on the other hand, is highly effective in such cases, being able to detect unseen authors, and to make use of this information to enhance the training data with additional examples. As a result, improvements provided by LAC-MR-ST range from 241.6% to 407.1%. Thus, LAC-MR-ST is not only able to reduce labeling efforts (as shown in the experiments with the DBLP collection), but it is also able to detect novel and important information (i.e., unseen authors), being highly practical and effective in a variety of scenarios.

In the next experiment, we used the DBLP collection to perform a comparison between LAC-MR-ST (Δ_{min}=0.7, ϕ_{min}=4), and the k-way Spectral Clustering algorithm [Han et al., 2005], when no training example is available. We adopted the evaluation methodology proposed in [Han et al., 2005], so that we can directly compare the performance of both algorithms. In this case, a confusion matrix is used to assess MicF$_1$ numbers. A different confusion matrix is associated with each ambiguous group, and the final performance is represented by the accuracy averaged over all groups.

Ambiguous Group	LAC-MR-ST	K-Way SC
A. Gupta	0.453	**0.546**
A. Kumar	**0.555**	**0.505**
C. Chen	0.365	**0.607**
D. Johnson	**0.710**	0.561
J. Martin	0.786	**0.939**
J. Robinson	**0.662**	**0.693**
J. Smith	**0.444**	**0.500**
K. Tanaka	**0.554**	**0.626**
M. Brown	**0.680**	**0.759**
M. Jones	**0.504**	**0.628**
M. Miller	**0.699**	**0.479**
Average	**0.583**	**0.622**

Table 5.15. MicF$_1$ numbers for DBLP collection.

Table 5.15 shows the results. Best results, including statistical ties, are highlighted

in bold. As it can be seen, both algorithms provide results that are statistically tied on almost all ambiguous groups. The K-way spectral clustering algorithm obtained superior performance on three ambiguous groups, while LAC-MR-ST was superior in one ambiguous group. It is important to notice that the k-way spectral clustering algorithm takes as input the correct number of clusters to be generated, that is, if there are k authors in a group, then this group is clustered into exactly k clusters [Han et al., 2005]. This is clearly unrealistic in an actual or practical scenario, but provides something closer to an upper-bound for an unsupervised algorithm that has privileged information. LAC-MR-ST, on the other hand, does not use this information, and works by detecting unseen authors, and incrementally adding new examples to the training data. Other point worth mentioning is that, as shown in Figure 5.12, with small labeling efforts, the performance of LAC-MR-ST is improved (greatly outperforming the unsupervised algorithm), demonstrating that LAC-MR-ST is cost-effective (the only exception is when only very few examples are available, because in this case it seems that LAC-MR-ST has some difficulties in detecting novel authors, hurting disambiguation performance).

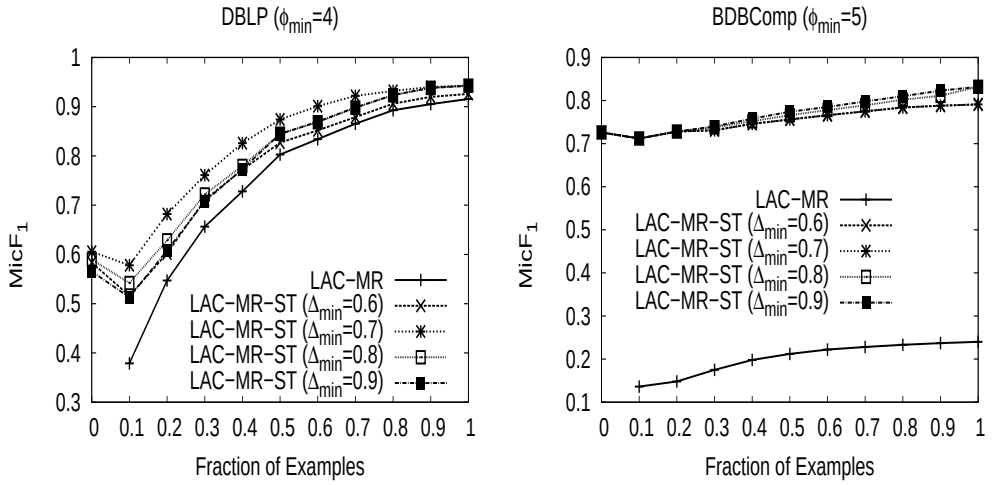

Figure 5.12. MicF$_1$ values for different Δ_{min} and ϕ_{min}.

5.4.4 Summary

In this section we have introduced a self-training demand-driven associative classification algorithm, LAC-MR-ST. It exploits the lack of evidence supporting any output in the training data to infer that a new output needs to be created. Further, it includes reliable predictions in the training data, so that new examples are automatically produced. As shown in our experiments, LAC-MR-ST is competitive to state-of-the-art

unsupervised algorithm. Further, LAC-MR-ST achieves higher classification performance when few examples are manualy provided.

5.5 Ordinal Regression and Ranking

Accurate ordering or ranking over instances is of paramount importance for several applications [Veloso and Meira, 2007; Veloso et al., 2007b, 2008a]. One clear application is Information Retrieval, where documents retrieved by search engines must be ranked according to the corresponding relevance to the query [Trotman, 2005]. Many features may affect the relevance of such documents, and, thus, it is difficult to adapt ranking functions manually. Recently, a body of empirical evidence has emerged suggesting that methods that automatically learn ranking functions offer substantial improvements in enough situations to be regarded as a relevant advance for applications that depend on ranking. Hence, learning ranking functions has attracted significant interest from the machine learning community. In the context of Information Retrieval, the conventional approach to this learning task is to assume the availability of examples (i.e., a training data, $\mathcal{S}$, which typically consists of document features and the corresponding relevance to specific queries), from which a learning function can be learned. When a new query is given, the documents associated with this query are ranked according to the learned function (i.e., this function gives a score to a document indicating its relevance with regard to the query). In this section it will be presented ranking algorithms based on demand-driven associative classification.

5.5.1 Related Work

Several methods have been proposed on how to compose a ranking function for Information Retrieval. More than one million possibilities to compute such functions were presented in [Zobel and Moffat, 1998]. Those possibilities take into account essentially a small number of features, such as term frequency, inverse document frequency, and document normalizations. Due to the growth in volume and popularity of the Web throughout the last decade, extra features have been proposed for improving retrieval, including those relative to the document structure (e.g., title, anchor text, and URL) and features concerning the importance of a document based on link analysis (e.g., Page Rank, HITS authority and hub). Thus, learning to rank methods that consider and combine all sorts of features for effective document retrieval and automatic ranking have become a topic of interest.

Several ranking algorithms based on machine learning techniques [Mitchell, 1997] have been proposed and applied for learning to rank in Information Retrieval. Ac-

cording to Cao et al. [2006], the current methods fall into three categories: (1) point-wise, (2) pair-wise, and (3) list-wise approaches. In the point-wise approach [Crammer and Singer, 2002; Nallapati, 2004], each training example is composed of a set of document features and its corresponding relevance relative to a query. The learning process tries to map features into relevance levels. In the pair-wise approach [Burges et al., 2005; Cao et al., 2006; Freund et al., 2003; Gao et al., 2005; Herbrich et al., 2000; Joachims, 2002; Qin et al., 2007; Tsai et al., 2007], each training example is composed of pairs of documents and the preference relation among them. In this case, the goal is to classify each pair into correctly or incorrectly ranked categories. Finally, in the list-wise approach [Cao et al., 2007; Xu and Li, 2007; Yue et al., 2007], a list of documents are used as training examples. A ranking function is learned, and then used to sort documents.

Nallapati [Nallapati, 2004] proposed a formalization of the ranking task as a binary classification problem (i.e., documents are assigned as relevant or irrelevant), exploring the use of classifiers such as SVM and Maximum Entropy. A discriminative model for ranking to optimize average precision was proposed in [Gao et al., 2005]. The Ranking SVM method, which is based on the pair-wise approach, was proposed in [Herbrich et al., 2000]. Joachims also applied SVM for learning ranking functions using click-through data for training [Joachims, 2002]. Other ranking algorithms based on SVMs include [Cao et al., 2006; Qin et al., 2007; Yue et al., 2007]. RankNet, which is an algorithm based on neural networks, was proposed in [Burges et al., 2005]. In [Tsai et al., 2007], RankNet was extended by proposing a fidelity loss function on the basis of the probabilistic ranking framework. RankBoost, which is a boosting algorithm for combining preferences, was proposed in [Freund et al., 2003]. Another boosting-based method is presented in [Xu and Li, 2007].

Other algorithms to produce ranking functions are based on Genetic Programming [Koza, 1992]. Fan et al. have proposed several algorithms for discovering ranking functions using Genetic Programming. In [Fan et al., 2004] an algorithm to automatically generate term-weighting schemes for different contexts (e.g., collections and users) was proposed. The work in [Trotman, 2005] presented another another ranking algorithm based on Genetic Programming. A combined component approach for generating ranking functions was proposed in [Almeida et al., 2007]. They use term-weighting components extracted from well-known ranking functions for discovering effective ranking functions.

5.5.2 Algorithm

In this section we present an algorithm which learns ranking functions for Information Retrieval, based on demand-driven associative classification. This algorithm will be referred to as LAC-MR-OR (standing for LAC-MR for ordinal regression).

LAC-MR-OR – Extending demand-driven associative classification algorithms to sort inputs is rather simple [Veloso et al., 2008a]. The first step is to extract from $\mathcal{S}$, decision rules associating inputs to outputs. In the context of Information Retrieval, inputs are documents features and outputs are relevance levels. These rules are extracted on a demand-driven basis, as described in Section 4.1. The next step is to calculate, using Equation 3.5, the likelihood of each relevance level (i.e., c_j) for each document (i.e., x_i).

Finally, the ranking position of input x_i can be estimated by a linear combination of the likelihoods associated with each output (or each relevance level), as shown in Equation 5.18. Higher values of $rank(x_i)$ indicates that input x_i should be placed in first positions of the rank. Basic steps of LAC-MR-OR are show in Algorithm 17.

$$rank(x_i) = \sum_{j=0}^{p} \big(c_j \times \hat{p}(c_j|x_i)\big) \tag{5.18}$$

Algorithm 17 Producing ranking scores using LAC-MR-OR.

Require: The training data $\mathcal{S}$, input $x_i \in \mathcal{T}$, σ_{min}
Ensure: $rank(x_i)$

1: $\mathcal{S}^{x_i} \Leftarrow \mathcal{S}$ projected according to x_i
2: $\mathcal{R}^{x_i} \Leftarrow$ rules $\mathcal{X} \rightarrow c_j$ extracted from $\mathcal{S}^{x_i}$, such that $\pi(\mathcal{X} \rightarrow c_j) \geq \sigma_{min} \times |\mathcal{S}^{x_i}|$
3: calculate membership probabilities $\hat{p}(c_j|x_i)$ for each output c_j
4: **r**eturn $\sum_{j=0}^{p} \big(c_j \times \hat{p}(c_j|x_i)\big)$

5.5.3 Empirical Results

In this section we empirically analyze the proposed algorithm, LAC-MR-OR. We first present the collections employed in the evaluation, and then we discuss the effectiveness of LAC-MR-OR in these collections.

The LETOR 3.0 Benchmark

LETOR [Liu et al., 2007] is a benchmark for research on learning to rank, released by Microsoft Research Asia[4]. It makes available seven subsets (OHSUMED, TD2003, TD2004, HP2003, HP2004, NP2003 and NP2004). Each subset contains a set of queries, document features, and the corresponding relevance judgments. Features cover a wide range of properties, such as term frequency, BM25, PageRank, HITS etc. Documents are given as inputs (i.e., x_i), and their relevance levels are the corresponding outputs (i.e., c_j). The goal is to place relevant documents in the first positions of the ranking. Pre-processing involved only the discretization [Fayyad and Irani, 1993] of attribute-values in $\mathcal{S}$.

Setup – In all experiments we used 5-fold cross-validation, and the final results of each experiment represent the average of the five runs. All the results to be presented were found statistically significant based on a t-test at the 95% confidence level.

Evaluation Criteria – Ranking performance is evaluated using NDCG@k, P@k, and MAP measures. Basically, these measures express the ability to place documents with high relevance in the first positions of the ranking. If the set of relevant documents for a query $q_j \in \mathcal{Q}$ is $\{x_1, \ldots, x_{m_j}\}$, and $\mathcal{D}_j$ is the set of ranked documents associated with query q_j, then P@k is defined in Equation 5.19 (where $r(x_i)$ is the true relevance of document $x_i \in \mathcal{D}_j$). Precision values are averaged over all queries.

$$\text{P@}k(\mathcal{D}_j) = \sum_{i=1}^{k} \frac{r(x_i)}{k} \tag{5.19}$$

For a single query, Average Precision (AP) is the average of the precision values obtained for the set of top k documents existing after each relevant document is retrieved, as given in Equation 5.20. MAP is obtained by averaging AP values over all queries, as shown in Equation 5.20.

$$\text{MAP}(\mathcal{Q}) = \frac{1}{|\mathcal{Q}|} \sum_{j=1}^{|\mathcal{Q}|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{P@}k(\mathcal{D}_j) \tag{5.20}$$

Finally, NDCG@k is defined in Equation 5.21, where $\mathcal{Z}_k$ is a normalization factor calculated to make it so that the NDCG value of a perfect ranking is 1. NDCG values are averaged over all queries.

[4]LETOR Web page: http://research.microsoft.com/users/LETOR/

$$\text{NDCG@}k(\mathcal{D}_j) = \mathcal{Z}_k \sum_{i=1}^{k} \frac{2^{rank(x_i)} - 1}{\log(1+i)} \tag{5.21}$$

Computational Environment – The experiments were performed on a Linux-based PC with a Intel Pentium III 1.0 GHz processor and 1 GB RAM.

Baselines and Parameters – Our evaluation is based on a comparison against state-of-the-art learning to rank algorithms such as R-SVM [Yue et al., 2007], FRank [Tsai et al., 2007], R-Boost [Freund et al., 2003], SVMMAP[Joachims, 2002], AdaRank[Xu and Li, 2007], and ListNet[Cao et al., 2007]. The ranking performance, as well as the corresponding parameters, for these algorithms are available at the LETOR Web page. For LAC-MR-OR, we set σ_{min}=0.005.

Analysis – Tables 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, and 5.22 show MAP numbers for the seven subsets. Best results, including statistical ties, are shown in bold. The result for each trial is obtained by averaging partial results obtained from each query in the trial. The final result is obtained by averaging the five trials. We conducted two sets of significance tests (t-test) on each subset. The first set of significance tests was carried on the average of the results for each query. The second set of significance tests was carried on the average of the five trials.

In five, out of seven subsets, LAC-MR-OR was the best overall performer, demonstrating the effectiveness of demand-driven associative classification. In most of the subsets, LAC-MR-OR achieved superior ranking performance when compared to the best baseline. The only exceptions occurred in HP2003 and HP2004 subsets, where AdaRank was the best performer. Still, LAC-MR-OR obtained a ranking performance which is much better than the performance obtained by the worst baselines. Gains provided by LAC-MR-OR range from 6.6% (relative to FRank in NP2003) to 42% (relative to FRank in TD2003).

Trial	LAC-MR-OR	R-SVM	R-Boost	FRank	ListNet	AdaRank	SVMMAP
1	**0.352**	0.304	0.332	0.333	**0.346**	0.344	0.342
2	**0.463**	0.447	0.445	0.438	0.450	0.446	0.454
3	0.460	**0.465**	0.456	0.456	0.461	**0.469**	**0.462**
4	**0.521**	0.499	0.508	0.513	0.511	0.514	0.518
5	**0.482**	0.453	0.464	**0.481**	0.461	0.471	0.450
Avg	**0.456**	0.433	0.441	0.444	0.446	0.449	0.445

Table 5.16. MAP numbers for OHSUMED subset.

Trial	LAC-MR-OR	R-SVM	R-Boost	FRank	ListNet	AdaRank	SVMMAP
1	0.169	0.164	0.110	0.113	**0.192**	0.153	0.172
2	0.293	0.258	0.291	0.297	**0.325**	0.251	0.237
3	0.365	**0.408**	0.251	0.155	0.381	0.290	0.342
4	**0.394**	0.236	0.262	0.212	0.275	0.322	0.276
5	0.219	**0.249**	0.222	0.238	0.202	0.125	0.196
Avg	**0.288**	0.263	0.227	0.203	0.275	0.228	0.244

Table 5.17. MAP numbers for TD2003 subset.

Trial	LAC-MR-OR	R-SVM	R-Boost	FRank	ListNet	AdaRank	SVMMAP
1	0.213	0.211	**0.247**	0.226	0.225	0.173	0.185
2	**0.276**	0.209	**0.281**	0.203	0.215	0.248	0.192
3	**0.285**	0.206	0.241	0.218	0.223	0.229	0.201
4	0.267	0.218	0.238	**0.285**	0.223	0.194	0.211
5	0.276	0.274	**0.299**	0.262	0.229	0.250	0.235
Avg	**0.263**	0.224	**0.261**	0.239	0.223	0.219	0.205

Table 5.18. MAP numbers for TD2004 subset.

Trial	LAC-MR-OR	R-SVM	R-Boost	FRank	ListNet	AdaRank	SVMMAP
1	**0.695**	0.625	0.685	0.591	0.593	0.621	0.623
2	**0.676**	0.662	0.666	0.645	0.648	0.620	0.640
3	0.670	0.695	0.711	0.673	**0.751**	0.660	0.714
4	0.751	**0.761**	0.733	**0.769**	0.724	0.702	0.736
5	0.748	0.735	0.743	0.642	0.732	**0.789**	0.721
Avg	**0.708**	0.695	**0.707**	0.664	0.689	0.678	0.687

Table 5.19. MAP numbers for NP2003 subset.

Trial	LAC-MR-OR	R-SVM	R-Boost	FRank	ListNet	AdaRank	SVMMAP
1	0.592	0.535	0.550	0.599	0.550	**0.700**	0.574
2	0.648	0.608	0.559	0.629	0.659	0.594	**0.669**
3	**0.870**	0.756	0.609	0.731	0.739	0.607	0.767
4	0.611	0.694	0.531	0.485	**0.728**	0.600	0.599
5	0.650	**0.701**	0.570	0.560	0.684	0.608	**0.701**
Avg	**0.675**	0.659	0.564	0.601	**0.672**	0.622	0.662

Table 5.20. MAP numbers for NP2004 subset.

Trial	LAC-MR-OR	R-SVM	R-Boost	FRank	ListNet	AdaRank	SVMMAP
1	**0.717**	0.684	0.634	0.674	0.728	0.715	**0.729**
2	0.808	0.796	0.813	0.804	**0.852**	**0.855**	0.775
3	0.737	0.783	0.781	0.737	**0.821**	0.801	0.785
4	**0.762**	**0.763**	0.745	0.684	**0.772**	0.752	0.719
5	**0.755**	0.679	0.692	0.648	0.657	0.732	0.579
Avg	0.756	0.741	0.733	0.709	0.766	**0.771**	0.717

Table 5.21. MAP numbers for HP2003 subset.

Trial	LAC-MR-OR	R-SVM	R-Boost	FRank	ListNet	AdaRank	SVMMAP
1	0.666	0.664	0.634	0.674	**0.728**	0.715	**0.717**
2	0.756	0.680	0.813	0.804	**0.852**	**0.855**	**0.845**
3	0.806	0.742	0.781	0.737	**0.821**	0.801	0.780
4	0.635	0.715	0.745	0.684	**0.772**	0.752	0.760
5	0.627	0.536	0.692	0.648	0.657	**0.732**	0.608
Avg	0.696	0.667	0.733	0.709	**0.766**	**0.771**	0.742

Table 5.22. MAP numbers for HP2004 subset.

We also evaluated LAC-MR-OR in terms of precision and NDCG. Figure 5.13 shows precision numbers obtained from the execution of LAC-MR-OR. LAC-MR-OR improved the precision at the first positions (it was always the best performer at P@1). Precision in the subsequent positions are similar to the precision achieved by the best baselines. NDCG numbers are shown in Figure 5.14. Only the best and the worst baselines are shown for comparison. Again, LAC-MR-OR showed some improvements at the first positions, and a performance which is similar to the one achieved by the best baselines in the subsequent positions. LAC-MR-OR outperformed the best baselines in five (out of seven) subsets. Again, AdaRank showed to be the best performer in HP2003 and HP2004 subsets.

5.5.4 Summary

In this section we introduced a learning to rank algorithm which is based on demand-driven associative classification. LAC-MR-OR is a simple extension of LAC-MR which makes a linear combination of the probabilities associated with each output. We evaluated LAC-MR-OR using information retrieval applications that depend on ranking. Our experiments suggest that LAC-MR-OR is currently one of the best learning to rank algorithms, providing gains that range from 6.6% to 42%.

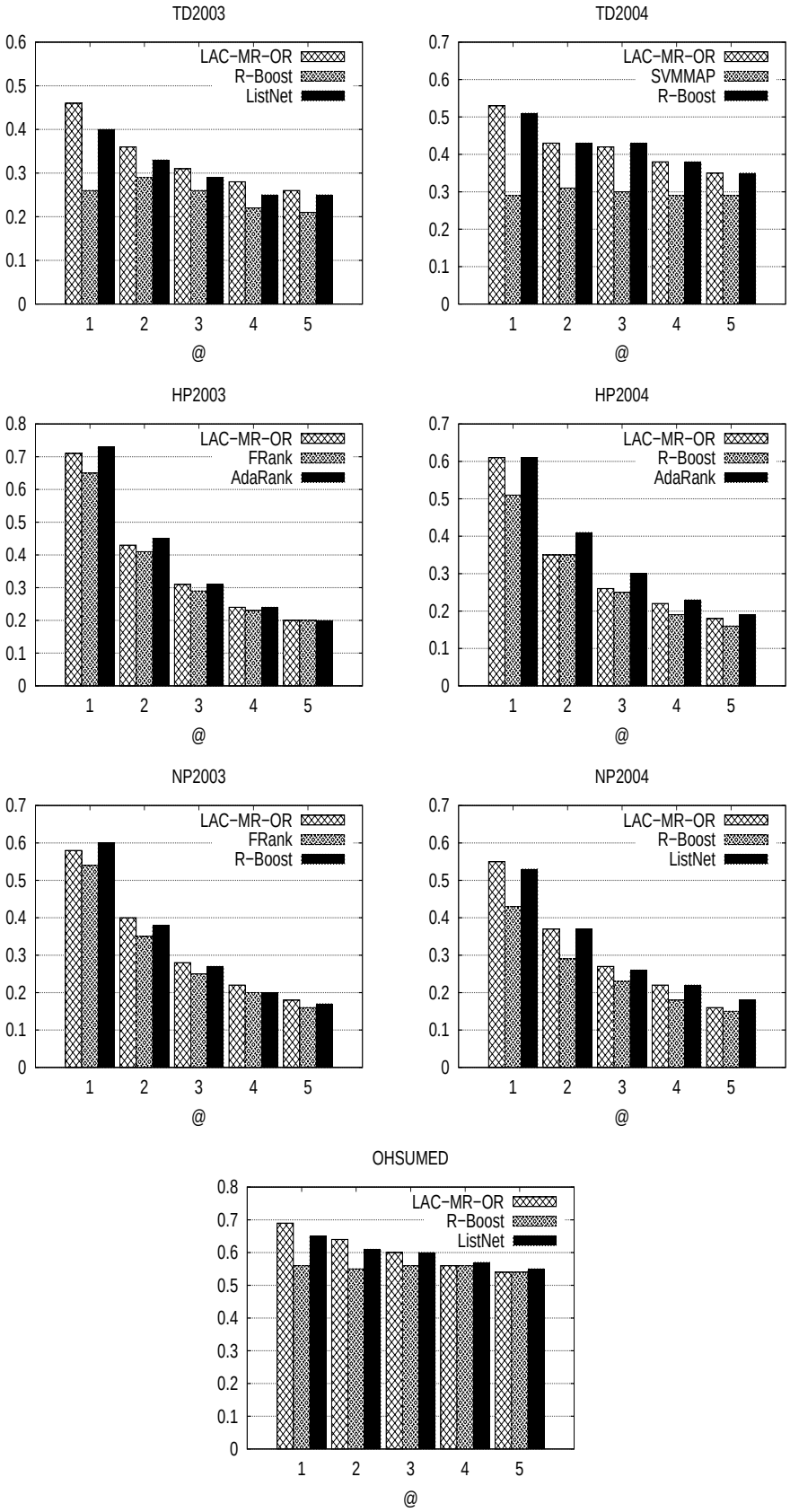

Figure 5.13. Precision numbers for different ranking algorithms.

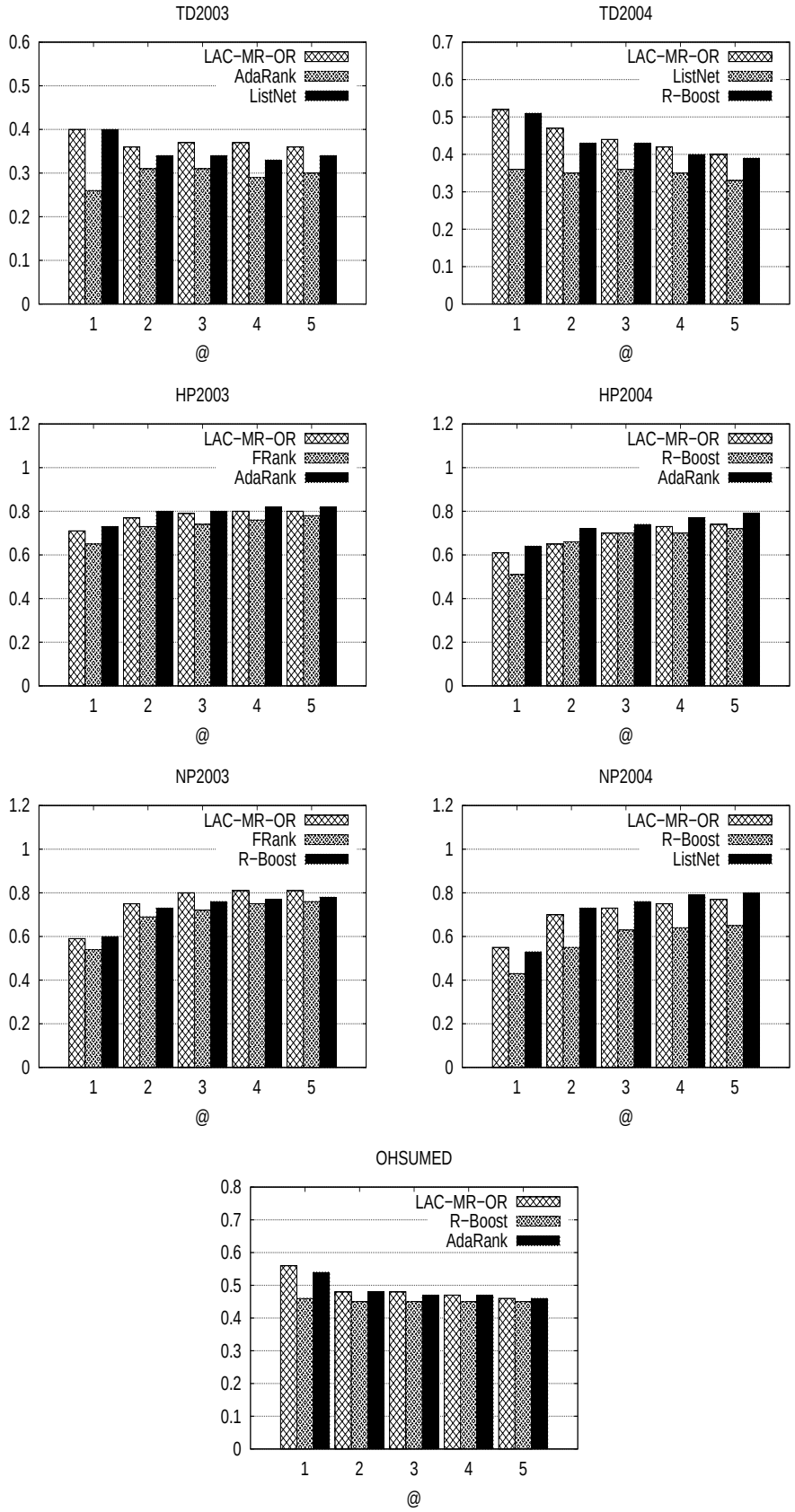

Figure 5.14. NDCG Numbers for different ranking algorithms.

Chapter 6

Conclusions

In this chapter we summarize the research contributions of this thesis and point out limitations and problems that remained open.

6.1 Summary of Results

The basic problem addressed in this thesis is known as classification, in which it is given a set of inputs and outputs, that are somehow related. The goal is to produce a mapping function that approximates this relationship, so that this function is used to predict the outputs for unknown inputs. We examined this problem from a function approximation perspective and proposed several classification algorithms. The first algorithm was shown to be PAC-efficient, and continuous improvements have led to the other algorithms proposed in this thesis. Such improvements resulted in demand-driven associative classification algorithms. We have shown that these algorithms produce functions that provide high classification performance in several real-world problems. The key insight that has led to these algorithms is that solving sub-problems may be much easier than directly solving the entire problem. Furthermore, we have shown that, frequently, each sub-problem demands approximation strategies that are very different from the strategy adopted when the entire problem is solved at once. Thus, producing approximation functions on a demand-driven basis may provide finer-grained results that are not achievable when the original problem is not broken into sub-problems. We successfully extended demand-driven associative classification algorithms to solve a number of problems that are related to the original classification problem, including cost-sensitive and cautious classification, multi-label classification, multi-metric classification, classification with limited labeling efforts, and ordinal regression. The relationship between the proposed algorithms is shown in Figure 6.1.

Apart from our results, we have implemented a vast amount of software, currently

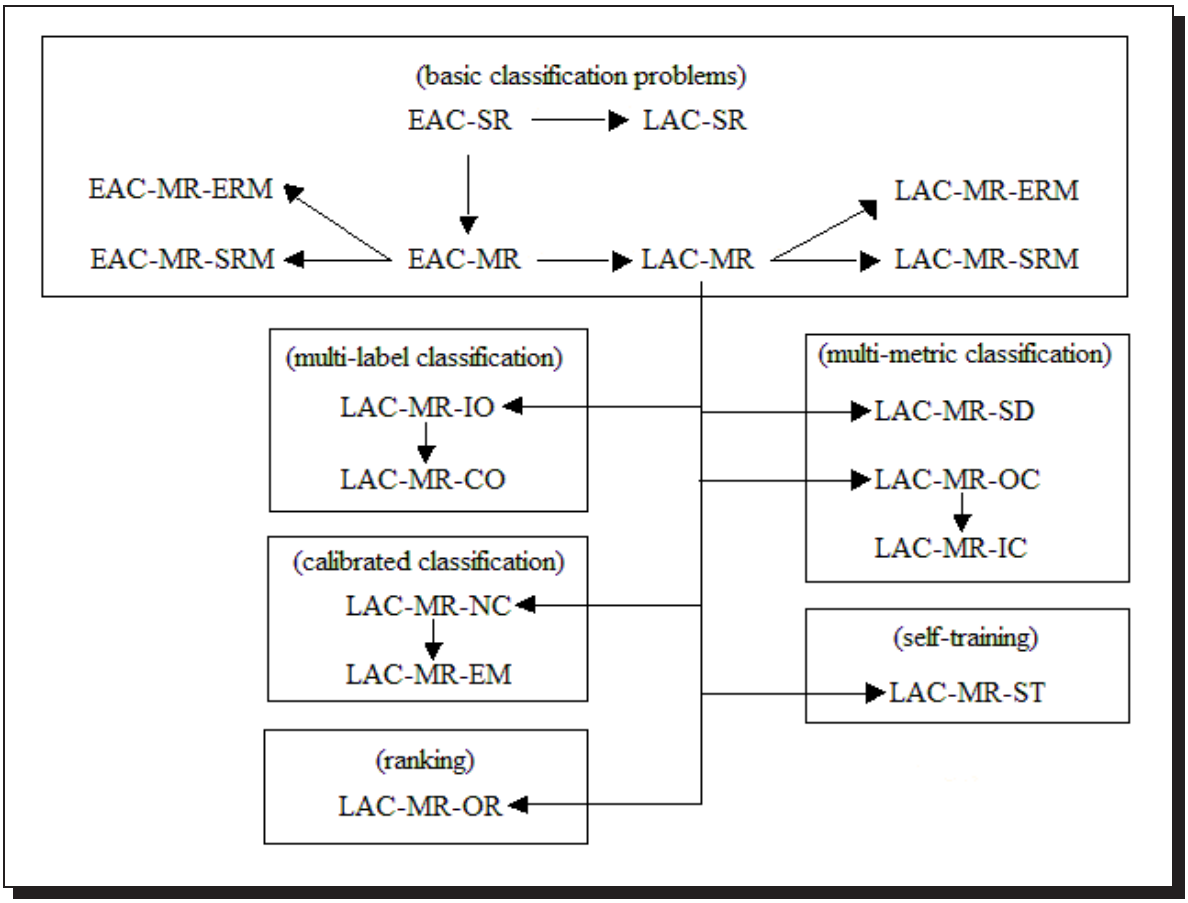

Figure 6.1. Relationship between the proposed classification algorithms.

available at `http://dcc.ufmg.br/~adrianov/software`. These implementations include:

- Demand-driven associative classification algorithms.
- Self-training associative classification algorithms.
- Multi-label associative classification algorithms.
- Algorithms that learn to rank.

6.2 Limitations

The proposed demand-driven associative classification algorithms have some limitations when compared to other classification techniques. These include:

1. Off-line discretization: the proposed algorithms are not able to process continuous attributes directly. First, these attributes must be discretized. We noticed

that the classification performance of the proposed algorithms depends on how effectively attributes are discretized. Supervised discretization techniques, such as [Fayyad and Irani, 1993], have shown to be effective in many cases. However, these techniques do not exploit the correlation among attribute-values, and this information can be lost. There are also discretization techniques that preserve the correlation between different attributes while producing the intervals [Mehta et al., 2005], but these are not supervised, and consequently, they may not consider important information about the correlation between inputs and outputs while producing intervals. Apart from this, discretization is a preprocessing step which many other classification algorithms (SVMs, KNN, many decision trees, etc.) do not need to perform.

2. Classification time: the proposed algorithms perform almost all the computation at classification time. While this strategy enables a great decrease in the total execution time, the classification time inevitably increases. We have shown that caching is extremely effective in keeping classification time low. However, this still may cause problems, for instance, in real-time applications, where adverse situations may take place, possibly increasing classification time.

6.3 Open Problems

Several interesting problems remained open. These problems include:

1. The choice of σ_{min}: in this thesis we have shown how to adapt cut-off values according to each sub-problem. However, the choice of the σ_{min} threshold that leads to the best classification performance is still an open problem. Interestingly, different values of σ_{min} induce nested classes of functions (i.e., functions derived from rules with support higher than σ_{min}). Thus, the same strategy we used to select the appropriate complexity of a function (i.e., Empirical/Structural risk minimization), can also be used to properly set σ_{min}.

2. Generalization bounds for associative classification algorithms: in this thesis we have used some general bounds derived from the stability or from the VC-dimension of a function. Specific bounds for associative classification algorithms can be derived using inputs in the test set. Specifically, we can confront how different features are associated to each other, in the training data and in the test set. The discrepancy observed between associations in the training data and associations in the test set may provide a powerful tool that can be used to bound generalization.

3. Semi-supervised associative classification algorithms: calibrated probabilities seems to be valuable in applications where few training examples are provided to the classification algorithm. During the classification of an input in the test set, if the probability associated with an output is substantially larger than the probabilities associated with the other outputs, then, in practice, the chance of misclassification is very low. In such cases, the input, along with the predicted output, can be incorporated to the training data (with low risk), increasing the number of examples and potentially improving classification performance. This is because, usually, features in inputs in the training data are redundantly sufficient to describe the examples, and thus associations between features of inputs in the training data and in the test set can be exploited.

4. Parallel associative classification algorithms: the use of large amounts of training data can enable the achievement of highly accurate mapping functions [Chan and Stolfo, 1993]. Thus, its is necessary the development of high-performance scalable classification algorithms, which are able to process large amounts of training data efficiently (which eventually may not fit in main memory [Zaki et al., 1999]). The algorithms proposed in this thesis can be greatly improved by parallel processing. First, each input in the test set induces a sub-problem, and each sub-problem can be processed independently from each other. This fact enables the use of a simple "bag os tasks" strategy, in which each sub-problem corresponds to a task. To make the process asynchronous, each processor will use a separate cache for storing its own decision rules. Load balancing is optimized, since processors will become idle only if no more tasks are available (i.e., there is no more inputs in the test set). Other degrees of parallelism can be further explored. For instance, the process of extracting rules from the training data can be efficiently parallelized following the strategies proposed in [Otey et al., 2003; Veloso et al., 2004; Otey et al., 2004].

Bibliography

Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proc. of the Int. Conf. on Management of Data (SIGMOD)*, pages 207–216. ACM Press.

Aha, D. (1997). Lazy learning. *Artificial Intelligence Review*, 11:1–5.

Aha, D., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Aizerman, M., Braverman, E., and Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837.

Almeida, H., Gonçalves, M., Cristo, M., and Calado, P. (2007). A combined component approach for finding collection-adapted ranking functions based on genetic programming. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 399–406. ACM Press.

Amsler, R. (1972). Application of citation-based automatic classification. Technical report, The University of Texas at Austin, Linguistics Research Center.

Angluin, D. (1992). Computational learning theory: survey and selected bibliography. In *Proc. of the Annual Symposium on Theory of Computing (STOC)*, pages 351–369. ACM Press.

Antonie, M., Zaïane, O., and Holte, R. (2006). Learning to use a learned model: A two-stage approach to classification. In *Proc. of the Int. Conf. on Data Mining (ICDM)*, pages 33–42.

Arunasalam, B. and Chawla, S. (2006). CCCS: a top-down associative classifier for imbalanced class distribution. In *Proc. of the Conf. on Data Mining and Knowledge Discovery (KDD)*, pages 517–522. ACM Press.

Baralis, E. and Chiusano, S. (2004). Essential classification rule sets. *Trans. on Database Systems*, 29(4):635–674.

Baralis, E., Chiusano, S., and Garza, P. (2004). On support thresholds in associative classification. In *Proc. of the Symposium on Applied Computing (SAC)*, pages 553–558. ACM Press.

Bennett, K. and Demiriz, A. (1998a). Semi-supervised support vector machines. In *Proc. of the Annual Conf. on Neural Inf. Processing Systems (NIPS)*, pages 368–374.

Bennett, K. and Demiriz, A. (1998b). Semi-supervised support vector machines. In *Proc. of the Annual Conf. on Neural Inf. Processing Systems (NIPS)*, pages 368–374.

Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 19–26.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled sata with co-training. In *Proc. of the Annual Conf. on Computational Learning Theory (COLT)*, pages 92–100. Springer.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Commun. ACM*, 36(4):865–929.

Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proc. of the Annual Conf. on Computational Learning Theory (COLT)*, pages 144–152. Springer.

Bottou, P. and Vapnik, V. (1992). Local learning algorithms. *Neural Computation*, 4(1):888–900.

Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research*, 2:499–526.

Boutell, M., Luo, J., Shen, X., and Brown, C. (2004). Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 89–96. ACM Press.

Calado, P., Cristo, M., Moura, E., Ziviani, N., Ribeiro-Neto, B., and Gonçalves, M. (2003). Combining link-based and content-based methods for web document classification. In *Proc. of the Conf. on Information and Knowledge Management (CIKM)*, pages 394–401. ACM Press.

Cao, Y., Xu, J., Liu, T., Li, H., Huang, Y., and Hon, H. (2006). Adapting ranking SVM to document retrieval. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 186–193. ACM Press.

Cao, Z., Qin, T., Liu, T., Tsai, M., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 129–136. ACM Press.

Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. In *Proc. of the European Conf. on Artificial Intelligence (ECAI)*, pages 147–149.

Chaitin, G. (1969). In the length of programs for computing finite binary sequences: Statistical considerations. *Journal of the ACM*, 16:145–159.

Chan, P. and Stolfo, S. (1993). Experiments on multistrategy learning by meta-learning. In *Proc. of the Conf. on Information and Knowledge Management (CIKM)*, pages 314–323. ACM Press.

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Available at `http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf`.

Chapelle, O., Scholhopf, B., and Kopriva, I. (2006). *Semi-Supervised Learning*. MIT Press.

Cheng, H., Yan, X., Han, J., and Hsu, C. (2007). Discriminative frequent pattern analysis for effective classification. In *Proc. of the Int. Conf. on Data Engineering (ICDE)*, pages 716–725.

Cheng, H., Yan, X., Han, J., and Yu, P. (2008). Direct discriminative pattern mining for effective classification. In *Proc. of the Int. Conf. on Data Engineering (ICDE)*, pages 169–178.

Clare, A. and King, R. (2001). Knowledge discovery in multi-label phenotype data. In *Proc. of the European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 42–53. Springer-Verlag Inc.

Cohen, I. and Goldszmidt, M. (2004). Properties and benefits of calibrated classifiers. In *Proc. of the European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 125–136. Springer-Verlag Inc.

Cohen, W. (1995). Fast effective rule induction. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 115–123. IEEE Computer Society.

Cohen, W. and Singer, Y. (1999). A simple, fast, and effective rule learner. In *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, pages 335–342.

Collobert, R., Sinz, F. H., Weston, J., and Bottou, L. (2006). Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712.

Comité, F., Gilleron, R., and Tommasi, M. (2003). Learning multi-label alternating decision trees from texts and data. In *Proc. of the Intl. Conf. on Machine Learning and Data Mining in Pattern Recognition (MLDM)*, pages 35–49. Springer-Verlag Inc.

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20(3):273–297.

Crammer, K. and Singer, Y. (2002). A new family of online algorithms for category ranking. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 151–158. ACM Press.

Cucker, F. and Smale, S. (2001). On the mathematical foundations of learning. *Bulletin (New Series) of the American Mathematical Society*, 39(1):1–49.

Cussens, J. (1993). Bayes and pseudo-bayes estimates of conditional probabilities and their reliability. In *Proc. of the European Conf. on Machine Learning (ECML)*, pages 136–152. Springer-Verlag Inc.

Dasarathy, B. (1990). *Nearest Neighbor Pattern Classification Techniques*. IEEE Computer Society.

DeGroot, M. and Fienberg, S. (1982). The comparison and evalution of forecasters. *Statistician*, 32:12–22.

Devroye, L. and Wagner, T. (1979). Distribution-free performance bounds for potential function rules. *Trans. on Inf. Theory*, 25(5):601–604.

Domingos, P. (1995). Rule induction and instance-based learning: A unified approach. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1226–1232.

Domingos, P. and Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–137.

Dong, G., Zhang, X., Wong, L., and Li, J. (1999). CAEP: Classification by aggregating emerging patterns. In *Proc. of the Int. Conf. on Discovery Science (ICDS)*, pages 30–49.

Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In *Proc. of the Annual Conf. on Neural Inf. Processing Systems (NIPS)*, pages 681–687. MIT Press.

Evgeniou, T., Pontil, M., and Poggio, T. (2000). Statistical learning theory: A primer. *International Journal of Computer Vision*, 38(1):9–13.

Fan, W., Gordon, M., and Pathak, P. (2004). Discovery of context-specific ranking functions for effective information retrieval using genetic programming. *Trans. on Knowledge and Data Engineering*, 16(4):523–527.

Fan, W., Zhang, K., Cheng, H., Gao, J., Yan, X., Han, J., Yu, P., and Verscheure, O. (2008). Direct mining of discriminative and essential frequent patterns via model-based search tree. In *Proc. of the Conf. on Data Mining and Knowledge Discovery (KDD)*, pages 230–238.

Fayyad, U. and Irani, K. (1990). What should be minimized in a decision tree? In *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, pages 749–754.

Fayyad, U. and Irani, K. (1993). Multi interval discretization of continuous-valued attributes for classification learning. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1022–1027.

Fern, X. and Brodley, C. (2003). Boosting lazy decision trees. In *Proc. of the Int. Conf. on Machine Learning, ICML*, pages 178–185.

Ferri, C., Flach, P., and Hernández-Orallo, J. (2004). Delegating classifiers. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, page 37. ACM Press.

Freund, Y., Iyer, R., Schapire, R., and Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *J. of Machine Learning Research*, 4:933–969.

Friedman, J., Hastie, T., and Tibishirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 2(38).

Friedman, J., Kohavi, R., and Yun, Y. (1996). Lazy decision trees. In *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, pages 717–724.

Fung, G. and Mangasarian, O. (2001). Proximal support vector machine classifiers. In *Proc. of the Conf. on Data Mining and Knowledge Discovery (KDD)*, pages 77–86.

Fürnkranz, J. and Flach, P. (2003). An analysis of rule evaluation metrics. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 202–209.

Gama, J. and Brazdil, P. (2000). Cascade generalization. *Machine Learning*, 45:315–343.

Gao, J., Qi, H., Xia, X., and Nie, J. (2005). Linear discriminant model for information retrieval. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 290–297. ACM Press.

Guyon, I., Boser, B., and Vapnik, V. (1992). Automatic capacity tuning of very large vc-dimension classifiers. In *Proc. of the Annual Conf. on Neural Inf. Processing Systems (NIPS)*, pages 147–155. MIT Press.

Han, H., Giles, C. L., Zha, H., Li, C., and Tsioutsiouliklis, K. (2004). Two supervised learning approaches for name disambiguation in author citations. In *Joint Conf. on Digital Libraries (JCDL)*, pages 296–305. ACM Press.

Han, H., Zha, H., and Giles, C. L. (2005). Name disambiguation in author citations using a k-way spectral clustering method. In *Joint Conf. on Digital Libraries (JCDL)*, pages 334–343. ACM Press.

Herbrich, R., Graepel, T., and Obermayer, K. (2000). *Large margin rank boundaries for ordinal regression*, chapter 7, pages 115–132. MIT Press.

Hilderman, R. and Hamilton, H. (2001). Evaluation of interestingness measures for ranking discovered knowledge. In *Proc. of the Pacific-Asia Conf. on Research and Development in Knowledge Discovery and Data Mining (PAKDD)*, pages 247–259. Springer.

Hutter, M. (2002). The fastest and shortest algorithm for all well-defined problems. *Int. Journal on Foundations of Computer Science*, 13(3):431–443.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 200–209. IEEE Computer Society.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proc. of the Conf. on Data Mining and Knowledge Discovery (KDD)*, pages 133–142. ACM Press.

Joachims, T., Cristianini, N., and Shawe-Taylor, J. (2001). Composite kernels for hypertext categorisation. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 250–257. ACM Press.

Kearns, M. and Vazirani, U. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

Kohavi, R., Sommerfield, D., and Dougherty, J. (1996). Data mining using MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*, pages 234–245.

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:4–7.

Kontkanen, P., MyllYmaki, P., Silander, T., and Tirri, H. (1998). Bayes optimal instance-based learning. In *Proc. of the European Conf. on Machine Learning (ECML)*, pages 77–88. Springer-Verlag Inc.

Koza, J. (1992). *Genetic Programming: On the programming of computers by natural selection*. MIT Press.

Kutin, S. and Niyogi, P. (2002). Almost-everywhere algorithmic stability and generalization error. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 275–282.

Lavrac, N., Flach, P., and Zupan, B. (1999). Rule evaluation measures: A unifying view. *Inductive Logic Prog.*, 1634:174–185.

Li, J., Dong, G., Ramamohanarao, K., and Wong, L. (2004). Deeps: A new instance-based lazy discovery and classification system. *Machine Learning*, 54(2):99–124.

Li, W., Han, J., and Pei, J. (2001). Efficient classification based on multiple class-association rules. In *Proc. of the Int. Conf. on Data Mining (ICDM)*, pages 369–376. IEEE Computer Society.

Liu, B., Hsu, W., and Ma, Y. (1998). Integrating classification and association rule mining. In *In Proc. of the Conf. on Data Mining and Knowledge Discovery (KDD)*, pages 80–86.

Liu, Y., Xu, J., Qin, T., Xiong, W., and Li, H. (2007). LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *L2R SIGIR Workshop*.

Mehta, M., Agrawal, R., and Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. In *Proc. of the Int. Conf. on Extending Database Technology (EDBT)*, pages 18–32.

Mehta, S., Parthasarathy, S., and Yang, H. (2005). Toward unsupervised correlation preserving discretization. *Trans. Knowl. Data Eng.*, 17(9):1174–1185.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

Mukherjee, S., Niyogi, P., Poggio, T., and Rifkin, R. (2006). Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Adv. Comput. Math.*, 25(1-3):161–193.

Nallapati, R. (2004). Discriminative models for information retrieval. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 64–71. ACM Press.

Niculescu-Mizil, A. and Caruana, R. (2005). Obtaining calibrated probabilities from boosting. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 413–420. AUAI.

Ortega, J., Koppel, M., and Argamon, S. (2001). Arbitrating among competing classifiers using learned referees. *Knowledge and Information Systems*, 3:470–490.

Otey, M., Parthasarathy, S., Wang, C., Veloso, A., and Meira, W. (2004). Parallel and distributed methods for incremental frequent itemset mining. *Teans. on System, Man and Cybernetics, Part B*, 34(6):2439–2450.

Otey, M. E., Wang, C., Parthasarathy, S., Veloso, A., and Meira, W. (2003). Mining frequent itemsets in distributed and dynamic databases. In *Proc. of the Int. Conf. on Data Mining (ICDM)*, pages 617–620.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers*, pages 61–74.

Poggio, T. and Girosi, F. (1998). A sparse representation for function approximation. *Neural Computation*, 10(6):1445–1454.

Poggio, T., Rifkin, R., Mukherjee, S., and Niyogi, P. (2004). General conditions for predictivity in learning theory. *Nature*, 428:419–422.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

Qin, T., Zhang, X., Wang, D., Liu, T., Lai, W., and Li, H. (2007). Ranking with multiple hyperplanes. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 279–286. ACM Press.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. M. Kaufmann.

Rahimi, A. and Recht, B. (2008). Uniform approximating functions with random bases. In *Allerton*, pages 43–49. SIAM.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465–471.

Rosasco, L., Vito, E. D., Caponnetto, A., Piana, M., and Verri, A. (2004). Are loss functions all the same? *Neural Computation*, 16(5):1063–107.

Schaffer, C. (1994). A conservation law for generalization performance. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 259–265. IEEE Computer Society.

Schapire, R. (1999). A brief introduction to boosting. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1401–1406. M. Kaufmann.

Schapire, R. and Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168.

Solomonoff, R. (1964). A formal theory of inductive inference. *Information and Control*, 7:1–22.

Stanfil, C. and Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM*, 13:7–34.

Tan, P., Kumar, V., and Srivastava, J. (2002). Selecting the right interestingness measure for association patterns. In *Proc. of the Conf. on Data Mining and Knowledge Discovery (KDD)*, pages 32–41. ACM Press.

Tresp, V. (2000). A bayesian committee machine. *Neural Computation*, 12(11):2719–2741.

Trotman, A. (2005). Learning to rank. *Information Retrieval*, 8(3):359–381.

Tsai, M., Liu, T., Qin, T., Chen, H., and Ma, W. (2007). FRank: a ranking method with fidelity loss. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 383–390. ACM Press.

Tsymbal, A., Pechenizkiy, M., and Cunningham, P. (2006). Dynamic integration with random forests. In *Proc. of the European Conf. on Machine Learning (ECML)*, pages 801–808. Springer-Verlag Inc.

Turing, A. (1951a). Can digital computers think? *A talk on BBC Third Programme, 15 May 1951.*

Turing, A. (1951b). Intelligent machinery, a heretical theory. *A lecture given to '51 Society at Manchester.*

Valiant, L. (1984a). A theory of the learnable. *Commun. ACM*, 27(11):1134–1142.

Valiant, L. (1984b). A theory of the learnable. In *Proc. of the Annual Symposium on Theory of Computing (STOC)*, pages 436–445. ACM Press.

Vapnik, V. (1991). Principles of risk minimization for learning theory. In *Proc. of the Annual Conf. on Neural Inf. Processing Systems (NIPS)*, pages 831–838. MIT Press.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley.

Vapnik, V. and Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280.

Veloso, A., Almeida, H., Gonçalves, M., and Meira, W. (2008a). Learning to rank at query-time using association rules. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 267–274. ACM Press.

Veloso, A., Ferreira, A., Gonçalves, M., Laender, A., Meira, W., and Belém, R. (2009a). Cost-effective on-demand associative
name disambiguation in bibliographic citations. *Trans. on Knowledge and Data Engineering*.

Veloso, A. and Meira, W. (2005). Rule generation and rule selection techniques for cost-sensitive associative classification. In *Proc. of the Brazilian Symposium on Databases (SBBD)*, pages 295–309.

Veloso, A. and Meira, W. (2006). Lazy associative classification for content-based spam detection. In *Proc. of the Latin American Web Congress LaWEB*, pages 154–161. IEEE Computer Society.

Veloso, A. and Meira, W. (2007). Efficient on-demand opinion mining. In *Proc. of the Brazilian Symposium on Databases (SBBD)*, pages 332–346. SBC.

Veloso, A., Meira, W., Cristo, M., Gonçalves, M., and Zaki, M. (2006a). Multi-evidence, multi-criteria, lazy associative document classification. In *Proc. of the Conf. on Information and Knlwledge Management (CIKM)*, pages 218–227. ACM Press.

Veloso, A., Meira, W., and de Carvalho, M. B. (2002a). Mining reliable models of associations in dynamic databases. In *Proc. of the Brazilian Symposium on Databases (SBBD)*, pages 263–277.

Veloso, A., Meira, W., de Carvalho, M. B., Rocha, B., Parthasarathy, S., and Zaki, M. (2002b). Efficiently mining approximate models of associations in evolving databases. In *Proc. of the European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 435–448. Springer-Verlag Inc.

Veloso, A., Meira, W., Ferreira, R., Guedes, D., and Parthasarathy, S. (2004). Asynchronous and anticipatory filter-stream based parallel algorithm for frequent itemset mining. In *Proc. of the European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 422–433. Springer.

Veloso, A., Meira, W., Gonçalves, M., and Zaki, M. (2007a). Multi-label lazy associative classification. In *Proc. of the European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 605–612. Springer-Verlag Inc.

Veloso, A., Meira, W., Macambira, T., Guedes, D., and Almeida, H. (2007b). Automatic moderation of comments in a large on-line journalistic environment. In *Proc. of the Int. Conf. on Weblogs and Social Media (ICWSM)*.

Veloso, A., Meira, W., Parthasarathy, S., and de Carvalho, M. B. (2003). Efficient, accurate and privacy-preserving data mining for frequent itemsets in distributed databases. In *Proc. of the Brazilian Symposium on Databases (SBBD)*, pages 281–292.

Veloso, A., Meira, W., and Zaki, M. (2008b). Calibrated lazy associative classification. In *Proc. of the Brazilian Symposium on Databases (SBBD)*, pages 135–149.

Veloso, A., Meira, W., Zaki, M., Gonçalves, M., and Mossri, H. (2009b). Calibrated lazy associative classification. *Information Sciences*.

Veloso, A., Meira, W., and Zaki, M. J. (2006b). Lazy associative classification. In *Proc. of the Int. Conf. on Data Mining (ICDM)*, pages 645–654. IEEE Computer Society.

Veloso, A., Rocha, B. G., de Carvalho, M., and Meira, W. (2002c). Real world association rule mining. In *Proc. of the British Nat. Conf. on Databases (BNCOD)*, pages 77–89. Springer-Verlag Inc.

Veloso, A., Zaki, M., Meira, W., and Gonçalves, M. (2009c). The metric dillema: Competence-conscious associative classification. In *Proc. of the SIAM Data Mining Conference (SDM)*. SIAM.

Veloso, A., Zaki, M., Meira, W., Gonçalves, M., and Mossri, H. (2009d). The metric dillema: Competence-conscious associative classification. *Statistical Analysis and Data Mining*.

Wang, J. and Karypis, G. (2005). HARMONY:efficiently mining the best rules for classification. In *Proc. of the SIAM Data Mining Conference (SDM)*, pages 205–216. SIAM.

Wang, K., Zhou, S., and He, Y. (2000). Growing decision trees on support-less association rules. In *Proc. of the Conf. on Data Mining and Knowledge Discovery (KDD)*, pages 265–269. ACM Press.

Witten, I. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.

Wolpert, D. (1995). The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework. *The Mathematics of Generalization*.

Wolpert, D. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1391.

Wu, Q. and Zhou, D. (2005). SVM soft margin classifiers: Linear programming versus quadratic programming. *Neural Computing*, 17(5):1160–1187.

Wu, X., Kumar, V., Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z., Steinbach, M., Hand, D., and Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37.

Xu, J. and Li, H. (2007). Adarank: a boosting algorithm for information retrieval. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 391–398. ACM Press.

Yang, Y. (1994). Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 13–22, Dublin, Ireland. ACM Press.

Yang, Y., Slattery, S., and Ghani, R. (2002). A study of approaches to hypertext categorization. *Journal of Intell. Inf. Systems*, 18(2–3):219–241.

Yin, X. and Han, J. (2003). CPAR: Classification based on predictive association rules. In *Proc. of the SIAM Data Mining Conference (SDM)*. SIAM.

Yue, Y., Finley, T., Radlinski, F., and Joachims, T. (2007). A support vector method for optimizing average precision. In *Proc. of the Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 271–278. ACM Press.

Zadrozny, B. and Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 609–616. IEEE Computer Society.

Zadrozny, B. and Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proc. of the Conf. on Data Mining and Knowledge Discovery (KDD)*, pages 694–699. ACM Press.

Zaki, M., Ho, C., and Agrawal, R. (1999). Parallel classification for data mining on shared-memory multiprocessors. In *Proc. of the Int. Conf. on Data Engineering (ICDE)*, pages 198–205.

Zhang, H. (2005). Exploring conditions for the optimality of naïve bayes. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 19(2):183–198.

Zhang, M. and Zhou, Z. (2007). ML-kNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

Zheng, Z. and Webb, G. (2000). Lazy learning of Bayesian rules. *Machine Learning*, 41(1):53–84.

Zhu, X. (2008). Semi-supervised learning literature survey. Technical report, University of Winsconsin, Computer Sciences, TR 150.

Zobel, J. and Moffat, A. (1998). Exploring the similarity space. *SIGIR Forum*, 32(1):453–490.